Article

Ladderpath Approach: How Tinkering and Reuse Increase Complexity and Information

Yu Liu [1,*], Zengru Di [1] and Philip Gerlee [2]

[1] International Academic Center of Complex Systems, Beijing Normal University, 519087 Zhuhai, China.
[2] Department of Mathematical Sciences, Chalmers University of Technology & University of Gothenburg, 40530 Gothenburg, Sweden.
* Correspondence: yu.ernest.liu@bnu.edu.cn

Abstract: The notion of information and complexity are important concepts in many scientific fields such as molecular biology, evolutionary theory and exobiology. Many measures of these quantities are either difficult to compute, rely on the statistical notion of information, or can only be applied to strings. Based on assembly theory, we propose the notion of a *ladderpath*, which describes how an object can be decomposed into hierarchical structures using repetitive elements. From the ladderpath two measures naturally emerge: the ladderpath-index and the order-index, which represent two axes of complexity. We show how the ladderpath approach can be applied to both strings and spatial patterns and argue that all systems that undergo evolution can be described as ladderpaths. Further, we discuss possible applications to human language and the origin of life. The ladderpath approach provides an alternative characterization of the information that is contained in a single object (or a system) and could aid in our understanding of evolving systems and the origin of life in particular.

Keywords: Complexity; Syntactic information; Shannon entropy; Kolmogorov complexity; Hierarchy; Module; Origin of life; Alien signal; Language; Evolution

Contents

Citation: Liu, Y.; Di, Z.; Gerlee, P. Ladderpath Approach: How Tinkering and Reuse Increase Complexity and Information. *Journal Not Specified* **2022**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

1. Introduction

In an interview, John Maynard Smith noted that while the 19th century had been the century of energy, in which science and engineering were concerned with transforming energy from one form to another (chemical to mechanical as in a steam engine or mechanical to electrical as in a dynamo), the 20th century was about information, and in particular the transformation of information [1]. This shift has been particularly pronounced in biology where the discovery of DNA as the carrier of hereditary information, and the subsequent sequencing of the entire human genome has shaped biology into a science of information.

Although most of us have an intuitive idea of what constitutes information, defining the term rigorously turned out to be challenging. This is partly due to the discrepancy in the meanings of the word *information*. In one sense information refers to knowledge or facts about the world (i.e., semantic information), whereas it can also refer to a characterization of the object structure that is independent of any interpretation or meaning (i.e., syntactic information) [2].

Consider the sentence: *Claude was an American mathematician, and Claude spent his childhood in Michigan.* The semantic information of this sentence is impossible to compute without properly defining the constituent words. We need to understand what “mathematician” refers to in the real world, which cannot be deduced from this sentence on its own. Therefore when considering an object in isolation the only sensible measure of information is a syntactic one.

When considering the transmission of signals along the communication channel, which was Claude Shannon’s motivation for formulating the information theory, we would like each transmitted symbol to be as informative as possible. However, when we turn our eye to the human language or to biology, we note that information is often conveyed by structures that contain redundancy.

These structures, which we often think of as complex, are somewhere between completely ordered (e.g., a crystal) and completely random (e.g., a perfect gas). From the intuitive point of view we would not like to assign a high complexity to a highly ordered sequence (e.g. a sentence with repetitive words), but on the other hand, we would not like to call a totally random sequence as complex since it lacks any structure. Instead, we would like to assign large complexity to sequences somewhere in between the repetitive and the random [3,4]. This requirement, motivated by our intuition, makes the definition of complexity challenging.

The first attempt to quantify complexity along these lines was made by Kolmogorov [5] and later refined by Chaitin [6–8], and is referred to as algorithmic complexity. It defines the complexity of a sequence as the length of the shortest possible computer program that could generate the sequence. This sounds promising, but due to the halting problem of the universal Turing machine, it has the practical drawback that there is no universal method for calculating the complexity of an arbitrary string [9,10]. Notwithstanding, algorithmic complexity could be approximated when defined on a weaker class of Turing machines (i.e., in particular contents or environments). For example, the well-known Lempel-Ziv complexity and algorithm that is widely used for file and data compression [11–13], related to an optimal rate of lossless string compression. Another example is the physical complexity that is particularly devised to characterize the amount of information about the surrounding environment that has been encoded in genomic sequences [14].

Another family of complexity measures build on Claude Shannon’s information theoretic concepts and in particular on Shannon entropy, which is defined as $H = -\sum p_i \log_2 p_i$, where the sum runs over all possible symbols in the sequence and p_i is the probability of observing symbol i (Shannon entropy itself is not a useful measure of complexity since it is maximized for completely random sequences) [15]. For example, Grassberger introduced a measure known as Effective Measure Complexity which is defined as the average amount of information contained in a string that can be used when guessing the next symbol [16]. A similar measure termed Statistical Complexity was introduced by Crutchfield and Young which measures the Shannon entropy of the probability distribution of the causal states present in the sequence [3]. In statistics, another related concept is Fisher information, representing the curvature of the relative entropy of the distribution of a random variable with respect to its parameters, which measures the information this random variable carries. It has a wide range of applications including measuring the complexity of learning tasks [17] and analyzing signals generated by the market [18].

Unlike the above-mentioned algorithmic complexity which is a form of absolute information of the individual object [19] and thus sensitive to the particular pattern of the sequence, the Shannon-entropy-related complexity measures rely on a statistical notion of information, which is insensitive to sequences. They can only be defined for sequences (or other objects) drawn from a statistical ensemble or equivalently sequences that are infinitely long. A solution to this problem is to assume that the underlying probability distribution for a specific sequence is uniform across all characters that appear in the sequence. However, this approach ignores the information (or "meaning") stored in the particular sequence. The question of how to characterize the syntactic information content and the complexity of finite length single instantiations thus remains open.

Since information is related to repeated patterns (as the Lempel-Ziv algorithm, for example, which utilizes repetitive substrings to compress information [20,21]), the above questions can also be approached from another angle, in a more general sense. The idea of joining or alternating objects that have already existed to construct new objects (e.g., gene sequences, molecules, technological inventions) was conceived in 1977 by François Jacob, phrased as "evolution as tinkering" [22]. This idea of accumulating information has laid the basis for many research fields such as protein interactions [23,24] and software modularity [25,26], and continues to inspire new research from a more general perspective, e.g. the evolution of network complexity where reuse plays a significant role [27]. In 1997 Donald Knuth extensively studied the "addition chain" in his famous computer science book, which has a similar motivation: The shortest addition chain for n can be considered as the most efficient addition sequence to reconstruct an integer from integers that have been constructed previously [28]. More recently, the concepts of "pathway complexity" [29] and "assembly space" [30,31] were developed to characterize the object complexity, by counting the number of construction steps, in which already built structures can be reused in subsequent constructing processes. This theory has been applied to detect biosignatures, and the hypothesis is that if a sufficiently complex molecule is found in abundance, biotic processes must have been involved [32]. Lately, the concept of "molecular assembly tree" was developed to characterize the hierarchical relationships within a group of distinct molecules, which shows great potential in fields such as drug discovery and origin of life research [33].

Following those lines above, i.e., "evolution as tinkering", "addition chain", and especially "assembly theory", in this paper we formalize an alternative approach to characterize information and complexity by focusing on the process of generating given objects, e.g. sequences, sentences, pictures, molecules, proteins, architectural structures. We call this the *ladderpath approach*. It is grounded not in the abstract theory of computation such as Kolmogorov complexity or Shannon entropy, but rather, as suggested by assembly theory, inspired by the tinkering process and reuse in construction that has shaped the living world.

In the following Section 2 we elaborate our *ladderpath approach* from scratch by giving related definitions one by one, followed by introducing an algorithm to calculate the ladderpath of an object. In Section 3 we relate the ladderpath to evolution, and lastly in Section 4 we discuss our findings in the context of interpreting unknown signals and ideas about life /evolution.

2. Ladderpath approach: starting from scratch

2.1. A thought experiment

In order to motivate the need for the definition of a ladderpath let us consider the following thought experiment: From somewhere in the universe, we have received a

sequence of letters (assuming we have somehow translated it into Latin letters):
ACXLGICXGOXEMZBRCNKXACXLPICXEMZBRCNKX.

Does this string contain any information, if yes, how could we understand or interpret it? First of all, we may say that this string contains information in the semantic sense, but we abandon this idea because we have no means of assigning a meaning to the individual letters in the string.

How do we approach the problem of extracting information from the string? We may start by searching for repetitions in the sequence. In the string, EMZBRCNKX appears twice. But in such a short string, the probability that this 9-letter substring appears twice from randomness alone is very low (if it appears 3, 4, 100 or more times, we may be more certain that it mustn't appear randomly, and must have a specific semantic meaning). Although appearing twice is not completely impossible, here we assume that anything that repeats twice or more is not random. Therefore we assume that EMZBRCNKX must have a specific semantic meaning. Nonetheless, by just looking at this string itself, we cannot infer what EMZBRCNKX really refers to; If we frequently see EMZBRCNKX appears together with other texts, pictures, objects, etc., we have the possibility to infer what it refers. But for now, we can only say that EMZBRCNKX must mean something or it must be a word or phrase in an alien language. In addition, we see ACX appears twice so it also means something.

How about other non-repetitive substrings such as LGIC and LPIC? Do they have particular meaning or semantic information? From this single string we can never figure it out. But if we find LPIC repetitively appears in some other places or sources, we can be sure that it must have a particular meaning. Thus, for the original string viewed in isolation, information in the semantic sense is just contained in EMZBRCNKX and ACX. [1]

2.2. Definition of generation-operation

Let us now move one step closer to a rigorous definition of a ladderpath. The motivation comes from the following question: Imagine we have a set that includes letters A, B, C, D, E, and F, and we need to obtain a specific string

$$\mathcal{X} = \text{ABCDBCDBCDCDEFEF}$$

then what is the quickest way (i.e., the least number of steps needed) to obtain $\mathcal{X}$? Surely, this question can be extended to other situations, e.g., we have a set of atoms and need to obtain a specific molecule (a preliminary idea in the case of molecules has been investigated in our previous publication [33]); or we have a set of hardware and need to obtain a bicycle; or we have a set of ingredients and condiments, and need to obtain a specific dish. Here we just take the string as a simple but generic example.

First of all, we define a ***basic set***, denoted as $\mathbf{S}_0 = \{\text{A}(0), \text{B}(0), \text{C}(0), \text{D}(0), \text{E}(0), \text{F}(0)\}$ which is defined to be a special type of *partially ordered multiset*, i.e., its elements are partially ordered (the ordered parts are separated by double slash "//", between which we can call "one ***level***"; and the unordered parts are separated by comma ","), and the number of instances (called *multiplicity*) of each element is written in the brackets behind the element. Note that the basic set $\mathbf{S}_0$ should be predefined, according to the specific research problem currently at hand, after which the analysis can be conducted (see Section 4.1 for more discussions). In this specific case we defined the basic set to be constituted from single letters.

Any element in this type of partially ordered multiset is called a ***building block*** (block for short). The elements in the basic set are called ***basic (building) blocks***. In addition, the string we want to obtain, $\mathcal{X}$, is called the ***target (building) block***. Note that, in $\mathbf{S}_0$, as shown above, the basic blocks are all unordered and their multiplicities are all zeros. Then, we define an operation on this type of partially ordered multiset:

- ***Generation-operation*** is defined as: take any number of blocks in the partially ordered multiset (the multiplicity decreases accordingly, but note that any block can be taken even if its multiplicity is zero or negative) and combine them in a certain way (note that the newly-generated blocks must not be present in the set), and then put the combined one back into the set at the level that is one level higher than the highest level of the constituted blocks. After this operation, a new partially ordered multiset of this type is obtained.

Note that: (1) It differs from the joining process in assembly theory [29,33] in that it allows any number of blocks to be joined and the putting-back step keeps the partial order. (2) The "negative multiplicity" is only a transient notation used for simplicity purposes, which does not appear in the final result. (3) Here "a certain way" could be different for different systems. For example, for the string system here, we define "a certain way" as: write the strings in the same line from left to right; For molecules, there could be a range of different generation-operations that could help capture properties like molecular isomerism (referring to Section 4.3 for more discussions). (4) "at the level that is one level higher than the highest level of the constituted blocks" means: for example, if the newly-generated block is made from blocks at level 1, level 3, and level 5, respectively, then the newly-generated block must be put at level 6.

For example, the operation of taking A and C in $\mathbf{S}_0$, combining them into AC, and then putting AC back is a generation-operation on $\mathbf{S}_0$, which can be denoted as $\mathbf{S}_0$:A+C=AC → $\mathbf{S}'$={ A(-1), B(0), C(-1), D(0), E(0), F(0) // AC(1) }. Note that AC is put back at the next level of A and C, indicated by the symbol "//".

For another example, the operation of taking D, D, F, AC in $\mathbf{S}'$, combining them into DDFAC and then putting it back is a generation-operation on $\mathbf{S}'$, denoted as $\mathbf{S}'$:D+D+F+AC =DDFAC → $\mathbf{S}''$={ A(-1), B(0), C(-1), D(-2), E(0), F(-1) // AC(0) // DDFAC(1) }. Because AC is at the highest level among D, F and AC which is level 2, so DDFAC is put back at level 3.

For a third example, the operation of taking B, C in $\mathbf{S}''$, combining them into BC and then putting it back is a generation-operation on $\mathbf{S}''$, denoted as $\mathbf{S}''$:B+C=BC → $\mathbf{S}'''$={ A(-1), B(-1), C(-2), D(-2), E(0), F(-1) // AC(0), BC(1) // DDFAC(1) }. Note that BC is put back at level 2, because B and C are at level 1.

Now, let's get back to the original question: how to obtain the target block $\mathcal{X}$. The following example (Ex1) shows several successive generation-operations after which we obtain $\mathcal{X}$:

Ex1. 1st generation-operation, $\mathbf{S}_0$:C+D=CD → $\mathbf{S}_1$={ A(0), B(0), C(-1), D(-1), E(0), F(0) // CD(1) }
. 2nd, $\mathbf{S}_1$:B+CD=BCD → $\mathbf{S}_2$={ A(0), B(-1), C(-1), D(-1), E(0), F(0) // CD(0) // BCD(1) }
. 3rd, $\mathbf{S}_2$:E+F=EF → $\mathbf{S}_3$={ A(0), B(-1), C(-1), D(-1), E(-1), F(-1) // CD(0), EF(1) // BCD(1) }
. 4th, $\mathbf{S}_3$:A+BCD+BCD+BCD+CD+EF+EF=$\mathcal{X}$
→ $\mathbf{S}_4$ = { A(-1), B(-1), C(-1), D(-1), E(-1), F(-1) // CD(-1), EF(-1) // BCD(-2) // $\mathcal{X}$(1) }
. Lastly, take one $\mathcal{X}$ out from $\mathbf{S}_4$, and then we achieve our goal: obtained one target block $\mathcal{X}$. The last step can be considered as one special generation-operation, that is, only take out but do not put back, which can be denoted as $\mathbf{S}_4$:$\mathcal{X}$(-1) → $\mathbf{S}_5$ = { A(-1), B(-1), C(-1), D(-1), E(-1), F(-1)//CD(-1), EF(-1) // BCD(-2) // $\mathcal{X}$(0) }

From $\mathbf{S}_0$ and $\mathbf{S}_1$ alone, we can infer that the generation-operation must be C+D=CD. In general, the same argument applies to $\mathbf{S}_i$ and $\mathbf{S}_{i+1}$. Therefore, we could simply write this path through which we achieve our goal as $\mathbf{S}_0 \to \mathbf{S}_1 \to \mathbf{S}_2 \to \mathbf{S}_3 \to \mathbf{S}_4 \to \mathbf{S}_5 \Rightarrow \mathcal{X}$. Evidently, there are many paths through which we can achieve the same goal. Here is another sequence of operations that also result in $\mathcal{X}$,

Ex2. 1st generation-operation, $\mathbf{S}_0$:D+B+C=DBC → $\mathbf{S}'_1$={ A(0), B(-1), C(-1), D(-1), E(0), F(0) // DBC(1) }
. 2nd, $\mathbf{S}'_1$:E+F=EF → $\mathbf{S}'_2$={A(0), B(-1), C(-1), D(-1), E(-1), F(-1) // DBC(1), EF(1)}
. 3rd, $\mathbf{S}'_2$:A+B+C=ABC → $\mathbf{S}'_3$ = { A(-1), B(-2), C(-2), D(-1), E(-1), F(-1) // DBC(1), EF(1), ABC(1) }
. 4th, $\mathbf{S}'_3$:DBC+DBC=DBCDBC → $\mathbf{S}'_4$={ A(-1), B(-2), C(-2), D(-1), E(-1), F(-1) // DBC(-1), EF(1), ABC(1) // DBCDBC(1) }
. 5th, $\mathbf{S}'_4$:C+D=CD → $\mathbf{S}'_5$={ A(-1), B(-2), C(-3), D(-2), E(-1), F(-1) // DBC(-1), EF(1), ABC(1), CD(1) // DBCDBC(1) }
. 6th, $\mathbf{S}'_5$:ABC+DBCDBC+D+CD+EF+EF=$\mathcal{X}$
→ $\mathbf{S}'_6$={ A(-1), B(-2), C(-3), D(-3), E(-1), F(-1) // DBC(-1), EF(-1), ABC(0), CD(0) // DBCDBC(0) // $\mathcal{X}$(1) }
. Lastly, $\mathbf{S}'_6$:$\mathcal{X}$(-1)
→ $\mathbf{S}'_7$={ A(-1), B(-2), C(-3), D(-3), E(-1), F(-1) // DBC(-1), EF(-1), ABC(0), CD(0) // DBCDBC(0) // $\mathcal{X}$(0) }

This path can be denoted as $\mathbf{S}_0 \to \mathbf{S}'_1 \to \mathbf{S}'_2 \to \mathbf{S}'_3 \to \mathbf{S}'_4 \to \mathbf{S}'_5 \to \mathbf{S}'_6 \to \mathbf{S}'_7 \Rightarrow \mathcal{X}$.

2.3. Definition of ladderpath

- For a path $\mathbf{S}_0 \to \mathbf{S}_1 \to \mathbf{S}_2 \to \cdots \to \mathbf{S}_n \Rightarrow \mathcal{X}$ through which we obtain the target block $\mathcal{X}$, we construct another partially ordered multiset in the following way: Take the final set $\mathbf{S}_n$, delete all of the blocks with zero multiplicity, and then set all other multiplicities to be the absolute value of the corresponding multiplicities (with the partial orders preserved). This procedure generates a new partially ordered multiset J, which we call the ***ladderpath*** of $\mathcal{X}$ that corresponds to this particular path.
- Any block in the ladderpath is called a ***ladderon***.

So, the ladderpath corresponding to Ex1 is

$$J_{\mathcal{X},1} = \{\mathrm{A, B, C, D, E, F}\ //\ \mathrm{CD, EF}\ //\ \mathrm{BCD}(2)\} \tag{1}$$

Note that we often omit the multiplicity "(1)" for simplicity. Likewise, the ladderpath corresponding to Ex2 is

$$J_{\mathcal{X},2} = \{\mathrm{A, B}(2), \mathrm{C}(3), \mathrm{D}(3), \mathrm{E, F}\ //\ \mathrm{DBC, EF}\} \tag{2}$$

Evidently, every path through which the target block is obtained corresponds to one ladderpath, but not vice versa. For example,

Ex3. 1st generation-operation, $\mathbf{S}_0$:D+B+C=DBC → $\mathbf{S}'_1$={ A(0), B(-1), C(-1), D(-1), E(0), F(0) // DBC(1) }
. 2nd, $\mathbf{S}'_1$:E+F=EF → $\mathbf{S}'_2$={A(0), B(-1), C(-1), D(-1), E(-1), F(-1) // DBC(1), EF(1) }
. 3rd, $\mathbf{S}'_2$:A+B+C+DBC+DBC+D+C+D+EF+EF=$\mathcal{X}$
→ $\mathbf{S}''_3$={ A(-1), B(-2), C(-3), D(-3), E(-1), F(-1) // DBC(-1), EF(-1) // $\mathcal{X}$(1) }
. Lastly, $\mathbf{S}''_3$:$\mathcal{X}$(-1) → $\mathbf{S}''_4$={ A(-1), B(-2), C(-3), D(-3), E(-1), F(-1) // DBC(-1), EF(-1) // $\mathcal{X}$(0) }

This path can be denoted as $\mathbf{S}_0 \to \mathbf{S}'_1 \to \mathbf{S}'_2 \to \mathbf{S}''_3 \to \mathbf{S}''_4 \Rightarrow \mathcal{X}$, and we can see that it also corresponds to ladderpath $J_{\mathcal{X},2}$. For a better understanding of the definition of the ladderpath, refer to Appendix B for more examples of the ladderpath of strings.

It is worth mentioning that

- Any target sequence has at least one ***trivial ladderpath*** in which all the ladderons are basic blocks.

Taking $\mathcal{X}$ as an example, its trivial ladderpath is $J_{\mathcal{X},0}$ = {A, B(3), C(4), D(4), E(2), F(2)}, which represents many paths that only use the basic blocks in any order to make $\mathcal{X}$.

The significance of defining a ladderpath lies in the fact that one ladderpath represents many different but equivalent paths that generate the target block, meaning that the ladderpath completely filters out all of the tedious information of the steps along paths. The information retained in a ladderpath (including ladderons and their multiplicities, and the order relationships among ladderons) completely but non-redundantly describes all of the equivalent paths. The ladderpath has several good properties:

- One ladderon is a block that has been reused (i.e., been part in a generation-operation at least twice). The reuse is the ultimate reason why we can simplify the process of making the target block (later we shall see that this type of simplification is closely related to the "information" contained in the target block).

- For any ladderpath $J_{\mathcal{X}}$,

$$N_a = \sum_{i \in J_{\mathcal{X}}} (m_i \times n_{i,a}) \tag{3}$$

holds for every letter, where N_a is the number of times the letter a appears in the target sequence $\mathcal{X}$, i represents all ladderons in this ladderpath $J_{\mathcal{X}}$, m_i is the ladderon i's multiplicity, and $n_{i,a}$ is the number of times the letter a appears in the ladderon i. Take $J_{\mathcal{X},1}$ as an example. The letter B appears 3 times in the target sequence $\mathcal{X}$, i.e. $N_{\mathrm{B}} = 3$. On the right hand side of Equation (3), the contribution of ladderon BCD is $2 \times 1 = 2$ (as its multiplicity is 2 and the letter B appears once in BCD); the contribution of ladderon B is $1 \times 1 = 1$ (as its multiplicity is 1 and the letter B appears once in B); and the contribution of other ladderons is zero. So the right hand side is also $2 + 1 = 3$. It is straightforward to see that Equation (3) holds for all other letters A, C, D, E and F, too. Likewise, we can also verify ladderpath $J_{\mathcal{X},2}$ and $J_{\mathcal{X},0}$.

Any ladderpath can be fully described by a partially ordered multiset (namely, the partially ordered multiset representation, e.g., Equation (1) and Equation (2)); Yet sometimes it is more intuitive to represent a ladderpath by a graph or network. We denote such a graph a ***laddergraph***. An example is shown in Figure 1a, where the levels represent the order relationships in the ladderpath, and the links among blocks represent generation-operations (e.g., E and F link up to EF, representing that EF is generated from E and F).

To make the laddergraph more concise, we make several conventions:

1. Dim all the lines that are linked to the basic blocks.
2. If a block i at a lower level can be linked to a block j at a higher level via other blocks, then even if i and j are directly linked, we do not draw the lines between i and j. For example, in Figure 1a, the block CD at a lower level is directly linked to the target block $\mathcal{X}$ at the higher lower since in the ladderpath $J_{\mathcal{X},1}$, CD is directly involved when generating $\mathcal{X}$; but because CD is linked to BCD, and BCD is linked to $\mathcal{X}$, then we should not draw a line between CD and $\mathcal{X}$.
3. The multiplicities of blocks can either be written down explicitly (as in Figure 1) or not.

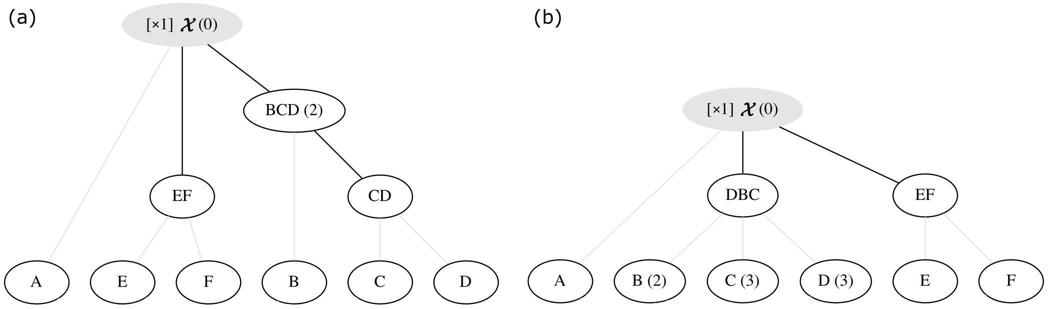


Figure 1. (**a**) The laddergraph that corresponds to ladderpath $J_{\mathcal{X},1}$. (**b**) The laddergraph that corresponds to ladderpath $J_{\mathcal{X},2}$. Grey blocks represent target blocks. "[×1]" in front of the target block represents that "we need to obtain 1 of such target block in the end". If (n) is added behind a block, it means the multiplicity (in the partially ordered multiset representation) of this block is n, while if there is no (n) behind, it means the multiplicity is 1. Finally, (0) behind the target block means that its multiplicity in this ladderpath is 0 (in principle, we should not draw blocks if their multiplicities are 0's, but here we explicitly drew the target block, just in order to show the readers the hierarchical relationships between it and other blocks).

In principle, the partially ordered multiset representation and the laddergraph representation of a ladderpath are one-to-one, yet because of the convention (2) and (3) above, the laddergraph representation may lose some information. Nevertheless, the most important information of the ladderpath is well-retained in the laddergraph, i.e., the hierarchical relationships among ladderons.

2.4. Definition of ladderpath-index (λ), order-index (ω), and size-index (S)

In this subsection, we will introduce three significant concepts: the ladderpath-index, and its associated concepts, size-index and order-index. But before that, we need to first define the "length unit of a ladderpath" and the "length of a ladderpath".

- For the string examples this paper describes, we define the ***length unit of a ladderpath*** as follows: for each operation, that concatenates any two strings (namely, blocks), we associate a unit called a "***lift***" (note that since how building blocks are combined together might be different in different systems, the length unit of a ladderpath could be defined differently, yet we always term the unit as a "*lift*").

Since each generation-operation actually corresponds to writing n blocks together, equivalent to repeating the action of "writing two blocks together" $(n-1)$ times. So, every generation-operation actually corresponds to a $(n-1)$ *lifts*. For example, in the path in Ex1 that $\mathbf{S}_0 \rightarrow \mathbf{S}_1 \rightarrow \mathbf{S}_2 \rightarrow \mathbf{S}_3 \rightarrow \mathbf{S}_4 \rightarrow \mathbf{S}_5 \Rightarrow \mathcal{X}$, the length of the 1st generation-operation C + D = CD is 1 *lift*, the length of the 2nd generation-operation B + CD = BCD is 1 *lift*, the length of the 3rd E + F = EF is 1 *lift*, and the 4th A + BCD + BCD + BCD + CD + EF + EF = $\mathcal{X}$ is 6 *lifts*. For the last special generation-operation, we can consider it as writing $\mathcal{X}$ and another hypothetical empty symbol together, and its length is thus 1 *lifts*. For anther example, in the path in Ex2, the length of each generation-operation is $2, 1, 2, 1, 1, 5, 1$ *lifts*, respectively. In the path in Ex3, the length of each generation-operation is $2, 1, 9, 1$ *lifts*, respectively.

As mentioned, the length unit of a ladderpath might be defined differently. For example, when studying chemical molecules, the length unit may be defined to be the formation of one chemical bond between atoms, ions, or molecular fragments; when studying evolution, the length unit may be defined to be the emergence of a new species, or a new physiological function; and so on. Therefore, the length unit of a ladderpath is a user-defined quantity. Nevertheless, as long as it is well-defined beforehand, it is not allowed to be altered in the following analyses.

We can now make the following definition:

- The ***length of a ladderpath*** J, denoted as $|J|$, is the sum of the lengths of all generation-operations along the ladderpath J. Note that, the length of any path is thus naturally defined as the sum of the lengths of all generation-operations along this path.

$|J|$ is to describe the "cost" that is required to generate the target block, associated with this ladderpath J. One ladderpath often corresponds to many paths that can generate the target block, but the lengths of all of these paths are identical (which is one of the convenient properties of a ladderpath). This implies that we can pick any of such paths to calculate $|J|$. For example, it is easy to verify that the length of the paths in Ex2 and Ex3 are both 13 *lifts*, and both paths indeed correspond to one ladderpath $J_{\mathcal{X},2}$ (evidently, we can conclude that $|J_{\mathcal{X},2}| = 13$ *lifts*). For the path in Ex1, its length is 10 *lifts*, so the length of its corresponding ladderpath $J_{\mathcal{X},1}$ is also 10 *lifts* (which is shorter than $J_{\mathcal{X},2}$).

We now define the "ladderpath-index":

- The ***ladderpath-index*** of target block $\mathcal{X}$, denoted $\lambda(\mathcal{X})$, is the length of the ***shortest ladderpath(s)*** of $\mathcal{X}$ (there may be one or several shortest ladderpaths). Thus, the length unit of ladderpath-index is also "*lift*".

Note that (1) ladderpath-index differs from assembly index [31,33] in that the latter is measured by counting the joining steps while the former is measured by the length of generation-operations where the length unit must be predefined for different systems. (2) Computing the shortest ladderpaths is not trivial at all. Although we have proved that it is at least as hard as an NP-complete problem (i.e., this problem cannot be solved in a polynomial time scale as the size of the problem increases), we were still able to develop a rigid procedure and algorithm, referring to Section 2.7 for details.

For the target block $\mathcal{X}$, $J_{\mathcal{X},1}$ is its shortest ladderpath indeed. So, $\mathcal{X}$'s ladderpath-index is 10 *lifts*, i.e., $\lambda(\mathcal{X}) = 10$ *lifts*. In fact, for any target block $\mathcal{X}$, its ladderpath-index $\lambda(\mathcal{X})$ describes one of its intrinsic properties, i.e., the smallest "cost" to generate $\mathcal{X}$. The shortest ladderpaths is a central concept since it correspond to the most compressed way of storing the information contained in $\mathcal{X}$, as we shall see soon.

We now proceed to the second critical concept, the "size-index":

- The ***size-index*** $S(i)$ of any block i (e.g., the target block or a ladderon) is the length of its shortest trivial ladderpath (there could be one or several shortest trivial ladderpaths, but they all have the same length).

If the basic blocks are single letters, the size-index is the number of letters in the string, e.g., size-index of $\mathcal{X}$ is 16 *lifts*, because its trivial ladderpath corresponds to the one where we generate $\mathcal{X}$ by just concatenating single letters.

The reason why the definition emphasizes the word "shortest" is that if the basic set includes not only single letters but also strings, the lengths of trivial ladderpaths might be different. For example, let the basic set be {A, B, C, D, E, F, BCD} for $\mathcal{X}$. The trivial ladderpath could then correspond to concatenating single letters as before, i.e., {A, B(3), C(4), D(4), E(2), F(2)} which has the length of 16 *lifts*. Alternatively, we could employ multi-letter strings in the basic set and obtain a trivial ladderpath of the form {A, BCD(3), C, D, E(2), F(2)} whose length is 10 *lifts*. In this case, we must choose the length of the shortest trivial ladderpath to be its size-index.

We now introduce the concept of "order-index":

- The ***order-index*** $\omega(\mathcal{X})$ of the target block $\mathcal{X}$ is defined to be

$$\omega(\mathcal{X}) := S(\mathcal{X}) - \lambda(\mathcal{X}) \tag{4}$$

  where $S(\mathcal{X})$ is the size-index of $\mathcal{X}$, and $\lambda(\mathcal{X})$ is the ladderpath-index of $\mathcal{X}$. Evidently, the unit of order-index is "*lift*", too.

This means that the order-index is equivalent to the number of *lift*s that are saved when generating the target block $\mathcal{X}$ via the shortest ladderpath compared to the trivial ladderpath, or in other words, the work that has been saved from combining blocks when constructing the target.

Last, we introduce an important property of the length of a ladderpath, which is crucial for calculating ladderpath-index and order-index:

- The length of a ladderpath $J_{\mathcal{X}}$ can be directly calculated from its partially ordered multiset representation, with no need to convert the ladderpath into one of its corresponding path. The length can be calculated as:

$$|J_{\mathcal{X}}| = S(\mathcal{X}) - \sum_{i \in J_{\mathcal{X}}} m_i \cdot (S(i) - 1) \tag{5}$$

  where $S(\mathcal{X})$ is the target block $\mathcal{X}$'s size-index, $S(i)$ is the ladderon i's size-index, m_i is the ladderon i's multiplicity, and i represents all ladderons in this ladderpath $J_{\mathcal{X}}$.

For example, in Ex1 the ladderpath was given by $J_{\mathcal{X},1}$ = { A, B, C, D, E, F // CD, EF // BCD(2) }. $\mathcal{X}$'s size-index is 16 *lifts*; the size-index of any basic block is evidently 1 *lift* (so their contributions are always 0, and thus no need to consider); the size-indices of CD, EF, BCD are $2, 2, 3$ *lifts*, respectively; so, $|J_{\mathcal{X},1}| = 16 - (2-1) - (2-1) - (3-1) \times 2 = 10$ *lifts*. Likewise, $|J_{\mathcal{X},2}| = 16 - (3-1) - (2-1) = 13$ *lifts*. As we can see, the lengths calculated in this method are identical with the ones calculated directly through the definition.

- In fact, based on Equation (5) and the definition of the order-index, the order-index of the target block $\mathcal{X}$ can be readily calculated from the following formula:

$$\omega(\mathcal{X}) = \sum_{i \in J_{\mathcal{X}}} m_i \cdot (S(i) - 1) \tag{6}$$

where i represents all ladderons in this shortest ladderpaths.

2.5. *Ladderpath-index and order-index are two axes of "complexity"*

The shortest ladderpaths of the target block $\mathcal{X}$ contains "the whole knowledge" about how to describe its complexity and the information it carries; The ladderpath-index and order-index calculated from the shortest ladderpaths are abstractions of this whole knowledge in different aspects:

- The ladderpath-index $\lambda(\mathcal{X})$ describes the amount of "information" that $\mathcal{X}$ carries, that is, how many extra steps/"costs" or how much extra "information" the external agent needs to input in order to generate $\mathcal{X}$, equivalent to the difficulties to reproduce $\mathcal{X}$ (which is distinct from the "information" that Shannon entropy or thermodynamic entropy refers to).
- The order-index $\omega(\mathcal{X})$ describes how much "information" can be saved, i.e., the amount of redundant "information" (equivalently, the difference between the trivial ladderpath and the shortest ladderpath). This is consistent with the intuition, as the more steps/"costs" (namely, how many *lift*s) it saves, the more ordered the target block $\mathcal{X}$ is.
- Now we can see that "complexity" that we often intuitively talk about has two aspects: One is described by the ladderpath-index λ which focuses more on the difficulties and costs of constructing the target; While the other aspect is described by the order-index ω which focuses more on how the target is built in an organized and hierarchical manner.
- Lastly, for a particular target (or target system), the sum of its ladderpath-index and its order-index is always equal to its size-index, which automatically solves the "normalization" problem that may occur when comparing different sized targets.

To be more clear, we consider a simple example: consider the sequence in Section 2 which had some structure, $\mathcal{X}$ = ABCDBCDBCDCDEFEF, and another random sequence with the same length, $\mathcal{W}$ = ABCDEFCFEDCBFDBA. Intuitively, the information that $\mathcal{W}$ carries is more than that carried by the non-random $\mathcal{X}$, because, to repeat $\mathcal{W}$, the amount of information needed (i.e., needed to be memorized) is larger than that needed to repeat $\mathcal{X}$. On the other hand, the random $\mathcal{W}$ is less ordered/organized compared to $\mathcal{X}$. These two aspects are intuitively consistent with the two concepts λ and ω: On one hand, $\lambda(\mathcal{W}) = 16 > \lambda(\mathcal{X}) = 10$, i.e., the information $\mathcal{W}$ carries is more than that $\mathcal{X}$ carries; on the other hand, $\omega(\mathcal{W}) = 0 < \omega(\mathcal{X}) = 6$, i.e., $\mathcal{W}$ is less ordered/organized than $\mathcal{X}$.

The concepts we have introduced apply not only to strings, but also to any other objects. Based on the definition of the order-index Equation (4), we can draw Figure 2a, where each diagonal is the contour line of the size-index S. We can see that under the same ladderpath-index (i.e., contains the same amount of information and is equally difficult), the more ordered an object is, the larger it is; while under the same size-index, the more ordered the object is, the smaller the ladderpath-index is, i.e., the less information it contains.

Now, we take a detailed example to further explain how the concept of ladderpath describes the "complexity" of objects. Consider the patterns shown in Figure 2b, which can be interpreted as stones placed in a clearing. These patterns can be described using the ladderpaths shown in Figure 2c (see Appendix C for details on how to compute the ladderpaths of these stone patterns), and from these the corresponding ladderpath-index and order-index can be computed.

Based on these we make the following observations:

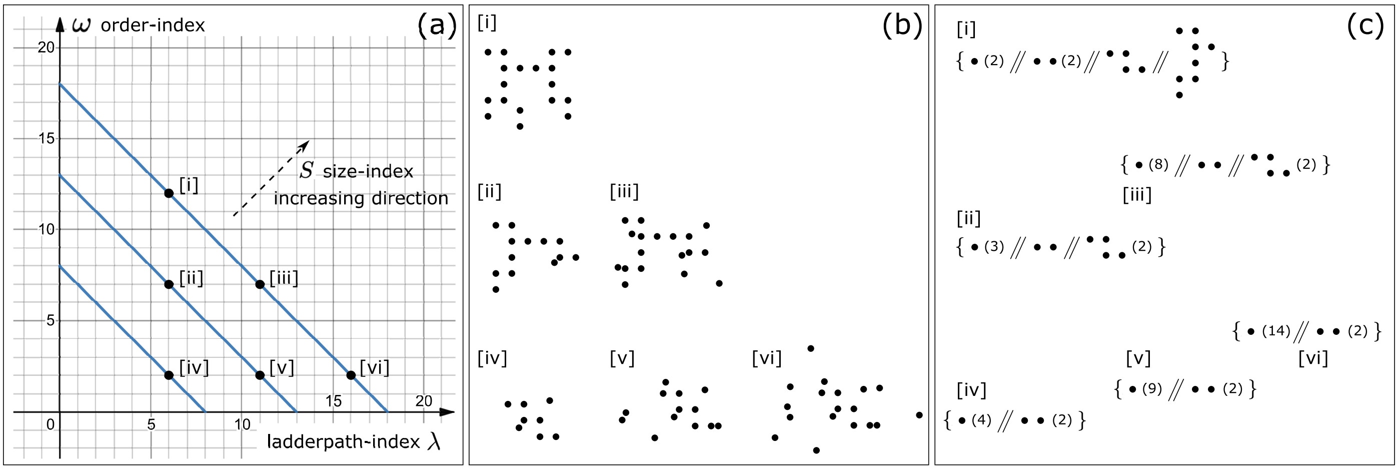


Figure 2. (**a**) The relationships among the ladderpath-index λ, the order-index ω and the size-index S. The blue diagonals are the contour lines of S. The points [i]-[vi] correspond to the patterns in (b) (note that one coordinate could correspond to an infinite number of patterns), and the coordinates are $(6,12), (6,7), (11,7), (6,2), (11,2), (16,2)$, respectively. (**b**) The patterns corresponding to the six coordinates in (a). (**c**) The ladderpaths of the six patterns.

1. For patterns with the same size-index (e.g., [i], [iii] and [vi]), as ω increases, the pattern becomes more and more ordered, which is consistent with our intuitions. On the other hand, as the ladderpath-index λ increases, the pattern becomes more and more difficult to reproduce (e.g., to reproduce [vi], we need to memorize the position of almost every stone, but to reproduce [i], we only need to memorize the description of the ladderpath, rather than the position of each stone), which is also consistent with our intuitions.
2. There are some counter-intuitive points indeed, yet they are also the most important: For example, [i] is more ordered than [ii], but the difficulty to generate either of them is the same, as their λ's are identical. [i] is more ordered than [iii] and [vi], but it is less difficult and takes fewer *lift*s to generate [i]. [i] is more ordered than [v], yet not only is it less difficult to generate [i], the size of [i] is also larger than [v] (the same argument applies to [ii], compared with [iv], [v] and [vi]; and so on). We shall see later that this point is the very key to explaining why the emergence of life is not as difficult as imagined before.

It deserves to mention that we may intuitively say that [vi] is more "complex" than [i], because the former looks more random /irregular /difficult to reproduce; while we may also say that [i] is more "complex" than [vi], because [i] needs more detailed, complex and delicate mechanisms to generate. However, it is not difficult to realize that these two "complex" refer to two distinct directions. The former is about how much information the system contains (or equivalently, how difficult, how much external information we need to input, to reproduce the system); while the latter is about how organized the system is. In fact, the two directions correspond to the ladderpath-index λ and the order-index ω, respectively. Therefore, now we are able to distinguish the two axes of "complexity".

The final remark on the two axes of complexity is that we always have $\lambda(\mathcal{X}) + \omega(\mathcal{X}) \equiv S(\mathcal{X})$, indicating that the two axes are not independent for a particular target or target system $\mathcal{X}$. It is true indeed, but in fact, any target can be placed in a particular position in these coordinates. This is because although the three indices are constrained by $\lambda + \omega \equiv S$, two of them are free. Referring to Figure 2, if we fix the size-index of a target, by rearranging the patterns of this target, it can move freely in the coordinates (e.g., imaging rearranging among pattern [i], [iii] and [vi]). It is exactly because of the internal structure and information of this target, its coordinates are fixed. Note that λ and S, or ω and S could be chosen as the axes, but our motivation here is to relate the intuition of complexity with the axes, and the intuition of complexity often comes from difficulty and order, as we have discussed above. To emphasize, the size is a significant factor of the complexity of an object, but it is not simply a proportional relationship—not simply the larger the size, the more complex—for example, referring to the Figure 2b, pattern [i] is larger than [v] but [i] is easier to be reproduced (as the ladderpath-index of [i] is smaller). This is exactly why we need the indices λ and ω to represent the two axes of complexity. With these concepts, we can now readily compare the complexity of objects with different sizes.

2.6. Extension of the concept: The ladderpath of a whole system

Up until now, we have discussed the ladderpath of a single target block, but all of these concepts can be extended to a whole system of blocks. To do that, we need to consider all of the blocks in the target as a whole and as one huge "target block". For example, if in the system, there are 2 strings of AAB, 3 strings of BC and 1 string of DDF, we can then consider them as one huge "target block": the string "AAB, AAB, BC, BC, BC, DDF", which we term the ***target system*** for convenience (but note that these strings are not connected, which is distinct from the single string AABAABBCBCBCDDF). It deserves to mention that the ladderpath of a target system is different from the concept of the "molecular assembly tree" which only handles a group of distinct types of molecules [33].

Now we take an example to illustrate the ladderpath of a target system (the extension of other concepts is then straightforward). Imagine the target system we need to obtain in the end is

$$\mathcal{Q} = \{\mathrm{ABDEDBED}(2), \mathrm{ABDED}, \mathrm{ABDABD}, \mathrm{CAB}(2), \mathrm{ED}(3)\}$$

while the basic set is $\mathbf{U}_0$ = {A(0), B(0), C(0), D(0), E(0)}. So, the following successive generation-operations make one path:

Ex4. 1st generation-operation, $\mathbf{U}_0$:A+B=AB → $\mathbf{U}_1$={ A(-1), B(-1), C(0), D(0), E(0) // AB(1) }
. 2nd, $\mathbf{U}_1$:C+AB=CAB → $\mathbf{U}_2$={A(-1), B(-1), C(-1), D(0), E(0) // AB(0) // CAB(1) }
. 3rd, $\mathbf{U}_2$:AB+D=ABD → $\mathbf{U}_3$={ A(-1), B(-1), C(-1), D(-1), E(0) // AB(-1) // CAB(1), ABD(1) }
. 4th, $\mathbf{U}_3$:ABD+ABD=ABDABD → $\mathbf{U}_4$={ A(-1), B(-1), C(-1), D(-1), E(0) // AB(-1) // CAB(1), ABD(-1) // ABDABD(1) }
. 5th, $\mathbf{U}_4$:E+D=ED → $\mathbf{U}_5$={ A(-1), B(-1), C(-1), D(-2), E(-1) // AB(-1), ED(1) // CAB(1), ABD(-1) // ABDABD(1) }
. 6th, $\mathbf{U}_5$:ABD+ED=ABDED
→ $\mathbf{U}_6$={ A(-1), B(-1), C(-1), D(-2), E(-1) // AB(-1), ED(0) // CAB(1), ABD(-2) // ABDABD(1), ABDED(1) }
. 7th, $\mathbf{U}_6$:ABDED+B+ED=ABDEDBED
→ $\mathbf{U}_7$={ A(-1), B(-2), C(-1), D(-2), E(-1) // AB(-1), ED(-1) // CAB(1), ABD(-2) // ABDABD(1), ABDED(0) // ABDEDBED(1) }
. Lastly, take all the blocks included in the target system $\mathcal{Q}$ from $\mathbf{U}_7$, and then the goal is achieved. The last step can be considered as one special generation-operation, i.e., take out but do not put back, denoted as $\mathbf{U}_7$:$\mathcal{Q}$(-1) → $\mathbf{U}_8$ = { A(-1), B(-2), C(-1), D(-2), E(-1) // AB(-1), ED(-4) // CAB(-1), ABD(-2) // ABDABD(0), ABDED(-1) // ABDEDBED(-1)}

Therefore, the ladderpath that corresponds to this path above is (its laddergraph representation is shown in Figure 3):

$$J_{\mathcal{Q}} = \{\mathrm{A}, \mathrm{B}(2), \mathrm{C}, \mathrm{D}(2), \mathrm{E} \mathbin{/\!/} \mathrm{AB}, \mathrm{ED}(4) \mathbin{/\!/} \mathrm{CAB}, \mathrm{ABD}(2) \mathbin{/\!/} \mathrm{ABDED} \mathbin{/\!/} \mathrm{ABDEDBED}\} \quad (7)$$

In fact, this path is $\mathcal{Q}$'s shortest ladderpath indeed (verified by our algorithm, detailed in Section 2.7). So, based on Equation (6), we are able to calculate $\mathcal{Q}$'s order-index $\omega(\mathcal{Q}) = 1 + 1 \times 4 + 2 + 2 \times 2 + 4 + 7 = 22$ *lifts*. As $\mathcal{Q}$'s size-index is $S(\mathcal{Q}) = 8 \times 2 + 5 + 6 + 3 \times 2 + 2 \times 3 = 39$ *lifts* (namely, the total number of letters included in all of the strings in $\mathcal{Q}$), we can then use Equation (4) to calculate its ladderpath-index $\lambda(\mathcal{Q}) = 39 - 22 = 17$ *lifts*. We can verify this result by looking at the total length of all the generation-operations in the corresponding path: The lengths of the first six generation-operations are all 1 *lifts*, the 7th generation-operation has the length of 2 *lifts*, and the last special generation-operation has the length of 9 *lifts*, so the answer is also 17 *lifts*.

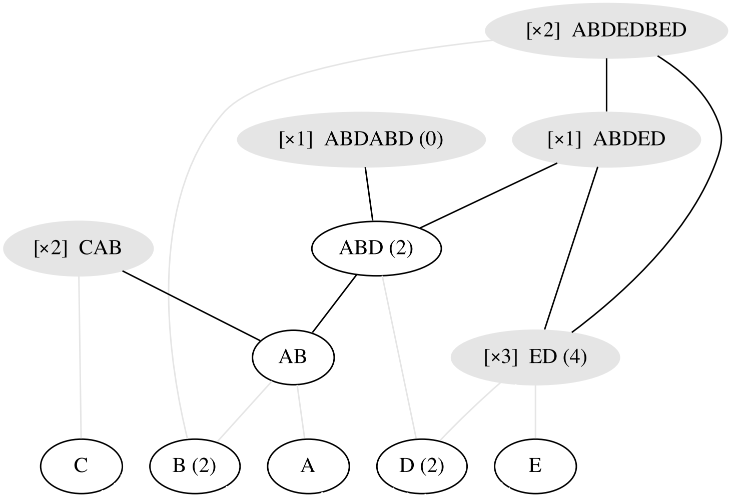


Figure 3. The laddergraph representation of one ladderpath of the target system $\mathcal{Q}$ (this ladderpath $J_{\mathcal{Q}}$ is actually the shortest one for $\mathcal{Q}$). All of the grey blocks constitute the target system $\mathcal{Q}$. "[$\times n$]" in front of the grey blocks represents that there are n such blocks included in the target system $\mathcal{Q}$. If (n) is added behind a block, it means the multiplicity (in the partially ordered multiset representation) of this block is n, while if there is no (n) behind, it means the multiplicity is 1. Finally, (0) means that its multiplicity in this ladderpath is 0 (in principle, we should not draw blocks if their multiplicities are 0's, but here we explicitly drew them, just in order to show the readers the hierarchical relationships among important blocks).

2.7. Algorithm to compute the shortest ladderpaths

To find the shortest ladderpath(s), we first developed the procedure that can give us one ladderpath each time, regardless of its length. Here we illustrate this procedure in the case of a target system instead of a target block, as the former is a more general case (without losing generalities, taking strings as an example):

1. First, create an empty multiset $\mathcal{H}$ to store blocks in. Later the ladderpath can be readily computed from $\mathcal{H}$.
2. Starting from the target system $\mathcal{Q}$, we preserve only one instance of each type of distinct blocks in $\mathcal{Q}$, and put all other repetitions into $\mathcal{H}$.
3. Keep slicing the blocks in $\mathcal{Q}$ (i.e. slicing strings into multiple substrings or letters) in a pre-determined systematic manner, until in $\mathcal{Q}$ there are at least two substrings (or letters) that are identical. Preserve only one of such identical substrings in $\mathcal{Q}$ and put all other repetitions into $\mathcal{H}$. Note that there could be many systematic manners to slice the string, referring to Appendix D for an example.
4. Repeat from step 3, until no repetitive substrings or letters can be found in $\mathcal{Q}$. Then, cut all of the remaining substrings into basic blocks (i.e. single letters in this case), and put them all into $\mathcal{H}$. Now, $\mathcal{H}$ records one ladderpath.
5. The final step is to align the (sub)strings in $\mathcal{H}$ in the right hierarchical order (that is, based on how they are sliced: which is the parent and which are the children), which results in the partially ordered multiset representation of this ladderpath.

See Appendix D for a detailed example. The output of the above procedure will depend on how we slice the strings. If we go through all deterministic slicing schemes we will generate all possible ladderpaths of the target system. From this collection of ladderpaths we select the shortest one. This gives us the algorithm to obtain the shortest ladderpaths. The code of this algorithm is available at `https://github.com/yuernestliu/ladderpath`.

The motivation here is to show a proof-of-concept algorithm and code to calculate the shortest ladderpath for a small target or target system, rather than providing an efficient and practical algorithm to deal with long strings or sequences (which might be used for string compression or gene sequence analysis for example). The idea is that if an algorithm for short strings can be specified, other algorithms for long strings, images, molecules, proteins, 3D objects, etc could be developed and sophisticated in the future.

Indeed, as Lempel-Ziv is also a lossless compression algorithm [12,13], it shares some similarities with this ladderpath algorithm in the aspects of general ideas of hierarchy and a few coding techniques. Nevertheless, besides their different motivations, the outputs of the two algorithms, resulting from their calculated slicing scheme respectively, are also different.

The final remark on the current version algorithm is that, it is not feasible to enumerate all ladderpaths for a large system as the number of possible ways to slice a string increases exponentially with its length. In fact, finding the shortest ladderpaths is at least as hard as an NP-complete problem, because it is a more complex version of the *addition chain* problem (first introduced by Knuth in his book [28]) which has been proven to be NP-complete [34].

Nevertheless, in practice it might be acceptable to just have "short-enough" ladderpaths. Then there might be shortcuts. For example, we can slightly change the above algorithm:

- In step 3, we bias the algorithm to search for repetitive substrings of maximum length.

In this way, although the ladderpath obtained is not guaranteed to be the shortest rigorously, it tends to be often short enough. The intuition is that the longer the ladderons (i.e. repetitive substrings) are, the more generation-operations are saved, and thus the shorter the resulted ladderpath is.

3. Ladderpath's significance in evolution: *ladderpath-systems*

A system's ladderpath-index λ corresponds to the least "cost" needed to generate the system. If we assume that the probability associated with each generation-operation is identical, then λ would be inversely proportional to the overall probability to generate this system, i.e., the larger λ is, the less probable the system can be eventually generated. For example, as shown in Figure 2, generating /reproducing pattern [i] is easier than generating /reproducing pattern [vi].

However, to safely reach this statement "generating [i] is easier than generating [vi]", a key assumption has to be made. This key assumption is: Any block that has been generated in previous steps can be reused in any amount, with no need to generate it from scratch (that is to say, the number of any block that has been generated is immediately infinite as if any of such block has an infinite reservoir; or equivalently, the number of ladderons is always infinite). For example, the last ladderon of pattern [i] (referring to Figure 2c, the pattern that consists of 8 stones) is reused once (i.e. repeated twice), meaning that the length of the trivial ladderpath of the last ladderon, namely 7 *lifts*, is saved. This kind of reuse is the very reason that makes generating [i] simpler than generating [vi].

In certain real-world scenarios, this type of reuse is possible (but not in every scenario). For example, in technology, if a new invention gets known in the society, it can be reused from that time on, with no need to reinvent it, that is, the reuse of ladderons is possible. But if one invents something but no other person knows it, the reuse of this invention would not be possible from the perspective of the whole society, meaning that the system (namely, the society) will not behave as the ladderpath theory describes.

Naturally, a further question arises: What conditions must a system satisfy such that it can be described by the ladderpath process? The answer is, we need two conditions: the first is the ability to generate new blocks, and the second is that blocks can replicate and thus increase in number. The first condition is implied in the generation-operation, while the second is implied in the assumption that the number of any ladderon is infinite. So,

- we call a system as a ***ladderpath-system*** if it satisfies the two conditions: (i) the ability to generate new blocks, and (ii) some or all blocks can replicate.

Intriguingly, these two conditions hold in many scenarios /systems, such as life, language, and technology.

Notice that for the first condition "the ability to generate new blocks", new blocks do not have to be generated by combining two existing blocks, that is, the generation-operation does not have to be exactly like the description in Section 2.2, and it can be defined differently according to the specific system in question. For example, if we consider single cells, not only gene recombination and horizontal gene transfer can be defined to be generation-operations (as the way before, i.e., by combining existing blocks), a genetic mutation can also be defined to be a generation-operation (in a different way, i.e., by changing an existing block). As for language, people create new words by combining existing words, borrowing words from other languages, and so on, resulting in new blocks generated. For technology, inventing is the way to generate new blocks which are always made by recombining or revising what already existed.

As for the second condition "replication", refers to either self-replication, or duplication with the aid of other things. That is, if a block can be categorized as a ladderon, it must have the ability to replicate. All ladderons are blocks, but only the blocks that are able to replicate become ladderons. Which blocks can replicate and which cannot set the stage for the "natural selection". One of the prominent properties of life is the ability to self-replicate: an individual and a species can self-replicate; and DNA can also replicate (it is actually the network consisting of DNA, RNA, and protein that can replicate [35–37]). For languages, the newly-created words and phrases can be used by other people is the way to replicate (when long-distance communication is not as convenient as today, the words that are newly created by local people hardly propagate in the whole society, so seeing from the whole society, these new words cannot become ladderons, which might be the mechanisms that a language is divided into different accents or evolves to different languages). For technology, if inventions are well-documented in papers, patents, etc., such that they can be reproduced, they can be considered as being able to replicate.

Now, that we have introduced the concept of ladderpath-systems, we may further address the following question: do totally random structures (white noise for example) carry the most information, or is the opposite true? There are two different angles to look at the "information" that the question refers to. (1) To repeat the particular sequence of a white noise, we need to memorize the whole sequence, which itself, in a sense, necessitates the great amount of explicit information. (2) Yet, complexity does not only mean the "difficulty to repeat" but also the amount of underlying hierarchical information that involves repetition and selection, which is another layer of information. We argue that the second layer of information is taken care of by the two properties of ladderpath-systems mentioned above. That is, the abundance of such selection-associated information is indicated by the ladderpath-index and the order-index both having a high value. Therefore, in this sense, white noise is almost devoid of such information, since only its ladderpath-index is high whereas its order-index is extremely low.

To end this subsection, the origin of life is likely to have occurred via this "ladderpath mechanism" (not in a "magical event"), that is, during the process lots of ladderons (which are able to replicate) must have been generated. In fact, there is evidence for that: e.g., obviously cells and species can self-replicate; the molecules that are made of individual cells or constitute chemical reaction networks (e.g. autocatalytic sets [35–37]) are indeed able to replicate.

4. Discussions

In this paper we have introduced the concept of a ladderpath of an object, which makes it possible to characterize the complexity of an object along two axes: the ladderpath-index and the order-index. We will now discuss how this construction plays out in two different context: ***isolated*** and ***non-isolated /united*** systems.

4.1. On information: alien signals (isolated system)

Let us imagine we have received a sequence $\mathcal{Y}$ of letters from another planet (some signals which have been converted to a sequence of letters):

$$\mathcal{Y} = \text{TBCDEFRBCDEFTEFHKREFHJKLMUVTEFPSMU}$$

and we need to figure out if it contains any kind of information. First of all, $\mathcal{Y}$ can be considered as an ***isolated system***, because there are no other signals or sequences that can be considered together with $\mathcal{Y}$.

By employing the ladderpath theory and the algorithm we have developed (Section 2.7), we find $\mathcal{Y}$'s shortest ladderpath

$$J_{\mathcal{Y}} = \{\text{T}, \text{B}, \text{C}, \text{D}, \text{E}, \text{F}, \text{R}(2), \text{H}(2), \text{K}(2), \text{J}, \text{L}, \text{V}, \text{T}, \text{P}, \text{S}, \text{M}, \text{U} \ // \ \ \text{EF}(2), \text{MU} \ // \ \text{TEF}, \text{BCDEF}\}$$

and its ladderpath-index is $\lambda(\mathcal{Y}) = 34 - 1 \times 2 - 1 - 2 - 4 = 25$ *lifts*. As a result, we can interpret $\mathcal{Y}$ as T [BCD [EF]] R [BCD [EF]] [T [EF]] H K R [EF] H J K L [MU] V [T [EF]] P S [MU], where the brackets separate ladderons.

Although it is too early to conclude that $\mathcal{Y}$ does not come from total randomness, we can at least see that the whole string is well-organized (we may infer that EF is the most important in the sentence as it appears the most number of times, while MU, TEF, and BCDEF are also important). In fact, $\mathcal{Y}$ is converted from the sentence (with punctuations and capitals neglected): *my mother made a chocolate cake your mother made a chocolate cake my chocolate cake was delicious your chocolate cake was not delicious I ate half of my chocolate cake and you ate half* (denoting this sentence as $\mathcal{Z}$ for convenience). Every letter in $\mathcal{Y}$ corresponds to a word in $\mathcal{Z}$. We can thus see that EF is *chocolate cake*, TEF is *my chocolate cake*, BCDEF is *mother made a chocolate cake* and MU is *ate half*.

There are four points needed to explain further. First of all, the ultimate reason that $\mathcal{Y}$ can be interpreted as $\mathcal{Z}$ is that $\mathcal{Z}$ is converted into $\mathcal{Y}$ in the first place on purpose. But in fact, replacing *chocolate cake* by any noun phrase consisting of two words makes sense (as well as replacing *mother* by *father*, etc). So, in principle, $\mathcal{Y}$ does not contain any information about *chocolate cake*, but only the relationships among the parts of the sentence. That is to say, we can never interpret EF, BCDEF, or any single letter if there is no information from other places (this is also why some old languages are uninterpretable).

Secondly, for those substrings that appear more than once, they only contain non-redundant information once. For example, the first part of $\mathcal{Z}$ can definitely be compressed as *my and your* [*mother made a chocolate cake*] (although there might be grammar errors, the meaning is clear), that is, the information contained in this part [*mother made a chocolate cake*] as a whole does not change at all, but just with a different qualifier ahead.

The third point is significant but also confusing: the reason why we are able to interpret those non-repetitive substrings is that we wrote $\mathcal{Z}$ in the first place and then converted it to $\mathcal{Y}$ on purpose. If we only look at $\mathcal{Y}$, there is no way we can tell if the letter that only appears once such as T and P is noise or contains useful information. Furthermore, after we replaced T by *my*, why we can interpret *my* as the actual meaning "of mine" is because the word *my* has already been repeated in my memory. A person who does not speak English does not have the word *my* repeated in his/her memory, so even if you tell him/her the sentence $\mathcal{Z}$, he/she cannot understand the meaning "of mine". Therefore, fundamentally, if a letter, a string, a word, a phrase, etc. has meanings, or equivalently, contains information, it must repeat, either in your memory/mind, in the system (the sentence) itself, or somewhere else.

Fourthly, there is an important assumption in the interpretation above: We considered the single letter as the basic unit of information, i.e., the basic set of the ladderpath consists of single letters. In fact, we do not have sufficient reasons to do that, because although it is reasonable to consider repetitive single letters (e.g., R, H, and K) as the basic blocks, it may not make sense for non-repetitive single letters such as P and S. In fact, we could consider PS as one basic block since P and S never appear independently. We are used to separating P and S because they are separated in our own language. What if in the alien language, they as a whole are just one letter (from the signals we received, we do not have enough reasons to believe that P and S are two separated letters)?

So, defining the basic set is the first step, which is subjective. We can define the basic set either based on some assumptions, hypotheses or facts from other sources, or only based on the target block itself. For $\mathcal{Y}$, we can define the basic set to be the set of individual letters as we already did above, yet we can also define it to be only the letters that are repeated. In the latter case, $\mathcal{Y}$ should be considered as $\mathcal{Y}'$ = T [BCD [EF]] R [BCD [EF]] [T [EF]] H K R [EF] H K [MU] [T [EF]] [MU], of which the shortest ladderpath is

$$J_{\mathcal{Y}'} = \{\mathrm{T}(2), \mathrm{R}(2), \mathrm{H}(2), \mathrm{K}(2), \mathrm{EF}(3), \mathrm{MU}(2), \mathrm{BCD} \mathbin{//} \mathrm{TEF}, \mathrm{BCDEF}\}$$

where BCD is considered to be one block since B, C and D always appear together. So, the length of $J_{\mathcal{Y}'}$, also the ladderpath-index of $\mathcal{Y}'$, is $\lambda(\mathcal{Y}') = 18 - 1 - 1 = 16$ *lifts* where 18 is the size-index of $\mathcal{Y}'$, the first 1 is the size-index of TEF (since EF is the basic block, we only need 1 *lift* to generate TEF), and the second 1 is the size-index of BCDEF (since BCD and EF are both basic blocks, we only need 1 *lift* to generate BCDEF). So, we can see that the ladderpath-index is different if the basic set is different (note that as long as the basic set is defined, it should not be changed in the following analyses). As for what basic set we should define, it is a different game. It should depend on what specific questions we are asking, which leads to the discussions in the next subsection.

4.2. On information: human language (non-isolated / united system)

Imagine we put the sentence $\mathcal{Z}$ (first mentioned in Section 4.1) in a human language (English in this case). Then what basic blocks shall we choose: single letters, single words, or only those repeated? The most straightforward answer is single words, because the single word is the basic unit that makes any sentence. In fact, there is no problem to define the basic block to be the single letter, but it is just more complicated, since there will be an extra level in the ladderpath, i.e., the level that makes words from letters. Yet in principle, the two ways have no real difference, but like measuring the same quantity in a different unit.

However, in the framework of English, it is inappropriate to define the basic set to be the set of only those repeated in the sentence, because although some word does not repeat in $\mathcal{Z}$ (e.g., *not*), it repeats in the framework of English, that is, it can appear in other possible English sentences. Therefore, when we discuss questions under the framework of English, we have actually considered the target sentence and all other possible English sentences as a whole (namely, a ***non-isolated /united system***) where even though a word did not repeat in the target sentence, it repeats in this non-isolated system, and we thus must consider these words as basic blocks.

The alien signal $\mathcal{Y}$ in Section 4.1 is an example of an isolated system, i.e., there are no other signals or sentences that can be considered together with $\mathcal{Y}$ to be a united /non-isolated system. Thus, the letters that are not repetitive in $\mathcal{Y}$ are not repetitive in the whole isolated system either (namely, the sentence itself), which is why we have no strong reason to consider these non-repetitive letters as basic blocks. Therefore, for the isolated case, we have two options, both mentioned in Section 4.1: After making some assumptions, take single letters as basic blocks as in $J_{\mathcal{Y}}$; Or only take repetitive letters as basic blocks and consider non-repetitive ones as noise, as in $J_{\mathcal{Y}'}$. Both approaches are correct, which are two interpretations of the original sequence. As for which approach reflects more of the reality, it is a totally different question, which should be addressed from other perspectives (referring to Appendix E for how we can apply this isolated /united-system idea to look for the evidence of intelligent life).

There is another point associated with the example in Section 4.1 that needs to be addressed: if we received many other signals of the same type as the signal $\mathcal{Y}$, then what should we do? In this case, we have to consider $\mathcal{Y}$ and all of the other signals we have received as a united system, then we need to analyze the ladderpath of this entire united system (whose size-index is evidently very high). If at the end we found that only BCDEF repeats once, then the calculated ladderpath-index would be very close to the size-index and the order-index would be very low, suggesting that the system is more likely to be random. Furthermore, what if there is a virtually infinite number of signals in principle but we only received one of them, which happened to be the $\mathcal{Y}$? In this case, we have to accept the fact that we can only consider $\mathcal{Y}$ alone as an isolated system, which may lead to a wrong or over-extrapolated conclusion (so extra evidence may be necessary, but that is an experimental problem outside of current considerations).

We can now come back to the question put forward in Section 1: We see that the "information" of a sentence in a language actually contains two levels of meanings. The first is the information in the narrow sense, which refers to the "information" we can extract from the united system consisting of the sentence itself and the language it belongs to, namely, the ladderpath. The second level of information is above the first level, referring to what reality, real object, behavior, etc that each ladderon corresponds to. The first level is about the sentence itself (namely the so-called syntactic information), while the second level is about the linkage between the sentence and the reality (namely the so-called semantic information).

4.3. On origins of life

As for how small the probability of the origin of life is, there is a famous metaphor called "junkyard tornado" [38]: The chance of emergence of life would be comparable to the chance that a tornado sweeping through a junkyard assembles a Boeing airplane. This is a vivid metaphor that is meant to help us comprehend how unlikely the emergence of life is. Yet it is misleading. The biggest flaw of this metaphor is that it suggests that the difficulty of making an airplane lies in assembling it with the most basic units (e.g., screws, semiconductors, or small elements lying in the junkyard), which is, however, not correct. Here we clarify this flaw.

How difficult is it for a person to make an airplane? First of all, let's simplify this question as much as possible: We assume that the airplane is just made of four parts, i.e. engine, propellers, wings, and control circuit.

A person living 10 000 years ago, would need to invent the four parts first, and then assemble them together. So the difficulty for them to make the airplane is the sum of the difficulty of inventing the four parts plus the difficulty of assembling. However, inventing all of the four parts altogether is easier than inventing each of them independently. For example, they need to invent metallurgy to obtain metals, but they only need to invent metallurgy once although all of the four parts require metals. The same reasoning applies to aerodynamics and wings and the propeller. Also, there are two wings, two propellers, and two engines, but he only needs to invent each of these parts once.

Therefore, the difficulty to make an airplane is not the sum of the difficulties of making each part individually, but that with repetitive parts excluded. This is the first level of meanings that the statement "making airplane is simpler than expected" implies.

The second level of meaning is that the basic set is different, that is, if the person lives in the 20th century all of the four parts are already matured technologies, but not the case for the ancients. Under the concept of ladderpath, for a modern person, the basic set consists of the engine, the propeller, the wing, and the control circuit, so an airplane's ladderpath-index (also corresponding to the difficulty) is just 4 *lifts*. But for the ancients, the basic blocks are merely ore and petroleum. Thus, the difficulty of making an airplane is much larger than 4 *lifts* in this case.

That is to say, in the 20th century, it is relatively easy for humans to invent airplanes, i.e., the probability of the emergence of airplanes is relatively large in the 20th century; while in ancient times, the probability of the emergence of airplanes is very small. The basic set the modern people face is the set derived from the set the ancient faces that has already undergone lots of generation-operations and generated lots of ladderons of higher levels.

The metaphor above about life actually compares life to the airplane, and compares atoms in the physical world to the scrap parts lying in the junkyard. Indeed, the probability that numerous atoms are assembled into a living system such as a cell by a simple physical process such as a tornado is extremely small. However, on one hand, life has a relatively low ladderpath-index (compared to its size-index) since it has many repeating elements (as we shall discuss below). On the other hand, life did not emerge all of a sudden from non-living systems, but probably through an elaborate path that generates ladderons, which gradually makes the system more and more complex, resulting in life at the end, like the "emergence" of the airplane in the 20th century.

Here we discuss the first point specifically: life has a relatively low ladderpath-index (compared to its size-index), due to its repetitive parts. First of all, ladderpath applies to any kind of object, including molecules. As for how to define the generation-operation and the length unit of ladderpath for molecules, it is different from the cases of strings. Here I give a reasonable scheme (there could be other schemes, see [33] for another attempt where molecules are considered to be composed of bonds, rather than atoms):

- The generation-operation for molecules is defined as follows: combine several molecular structures or fragments (namely, the blocks) into one, whereas the combination means that chemical bonds between atoms are formed.
- The length unit of a ladderpath is defined to be one chemical bond formed. That is, if n chemical bonds are formed in one generation-operation, the length of this generation-operation is n *lifts*.

Note that there is a range of possible ways in which molecular fragments could be combined. So, generation-operations could have different types (as we have mentioned after the definition of generation-operation in Section 2.2). For example, CH_3CHCH_2, CH_3 and H could be combined into $CH_3CH_2CH_2CH_3$ (namely, butane) or $HC(CH_3)_3$ (namely, isobutane). The choice of generation-operations should be recorded along with the calculated ladderpath as the necessary context.

For lots of molecules, the difficulties of making them (i.e., one aspect of complexity that is described by the ladderpath-index) are smaller than expected ("expected" refers to "assembling individual atoms to make a molecule"). For example, NADH has two ribose groups, so NADH's ladderpath-index is smaller than the number of atoms it contains; The backbone of RNA is made of lots of identical ribose rings and phosphate groups, and many types of proteins (such as tetrameric proteins) are made of repetitive structures, so their ladderpath-indices are smaller than the number of constituted atoms; Gene sequences have many repetitive segments, so their ladderpath-index is also smaller than the number of constituted base pairs. Furthermore, if we consider a group of molecules altogether (i.e., in the case of a target system, instead of an individual target block), the ladderpath-index will be smaller than the sum of individual ladderpath-indices. For example, if we consider 1 *mol* of NADH, 1 *mol* of ATP and 1 *mol* of FAD together, lots of their structures are repetitive.

We often say a protein is complex because we are wondering that although it contains so many atoms, why it is still so "ingeniously designed" [39]. But after we extract the repetitive structures of proteins, its "complexity" is much smaller (more precisely, its ladderpath-index is relatively small, compared to its size-index). The same argument applies to life. As we consider a living system (e.g., a cell) as a whole, its complexity is much smaller than the sum of the complexities of each individual part. Although it could still be very complex, it makes the emergence of life look easier.

4.4. On why life is ordered

Let's begin with an observation. For a ladderpath-system (namely, any system that satisfies "being able to generate new blocks" and "replication"), each time a new ladderon is generated, the ladderpath-index of the whole system is increased (equivalently, the information the whole system contains is increased); each time a ladderon is replicated, the order-index of the whole system is increased. Therefore, the ladderpath-index and the order-index of a ladderpath-system increase naturally, i.e., a ladderpath-system naturally evolves towards “complex”, which is an inevitable consequence of its two essential properties: “being able to generate new blocks” and “replication”.

Now, before we ask *why life is ordered*, let’s first ask: How do we interpret life is ordered, equivalently, well-organized? It is an intuitive thesis easily obtained from ordinary observations. We have two ways to interpret “ordered” here. Taking a cell as an example, the first interpretation is straightforward: it means a vast number of atoms organized into large biomolecules, which are organized into organelles, and in turn into a cell.

The second interpretation is that the atoms constituting the cell are unable to spread evenly in its space, and they are restricted at the positions, energy-levels and states of the large biomolecules and organelles (which is an interpretation in the sense of thermodynamics). As a result, the total number of all possible states in the state space of the cell is much smaller than that of an ideal gas (a typical example in thermodynamics) of the similar size. This implies that the cell’s thermodynamic entropy is low (referring to Appendix F for the discussion on the connections between the ladderpath and Shannon entropy).

The first interpretation refers to “ordered in the sense of the ladderpath”, while the second interpretation clearly refers to “ordered in the sense of thermodynamic entropy”. We can see that the two interpretations of “ordered” are not equivalent, although either makes sense in its own perspective.

Depending on how we interpret “ordered” we will approach the question *Why is life ordered?* very differently. Under the concept of ladderpath, this question might be solved readily, because evolving towards more and more ordered states is an intrinsic property of a ladderpath-system (to which life belongs), determined by the underlying evolution mechanism as discussed before. Yet, under the framework of thermodynamic entropy, this question is still a bit mysterious, because according to the second law of thermodynamics the entropy of a closed system spontaneously increases. Although, as an open system, the decrease of life’s entropy does not violate the second law, we still do not understand why/how life organizes itself so that it can resist the increase of thermodynamic entropy (some authors used more general /vague but more intriguing statements [40,41], e.g., “the entropy of life tends to decrease”, “life feeds on negative entropy”, etc., but all of these statements are basically equivalent).

If the following two statements are true, that is, “life must be a ladderpath-system (or put differently, life organizes itself into a ladderpath-system)” and “a ladderpath system is able to resist the increase of thermodynamic entropy”, then the question “why/how life organizes itself so that it can resist the increase of thermodynamic entropy” is solved. While it has been demonstrated above that life is a ladderpath-system, we thus need to figure out whether a ladderpath-system is able to resist the increase of thermodynamic entropy. The answer is likely to be yes, but a full proof /demonstration could not be provided here, as it requires much more work, both theoretically and empirically. Nonetheless, at least this type of question such as *Why is life ordered?* has been pinned down to a better understanding of ladderpath-systems. Since ladderpath-systems have a rigid mathematical definition, these questions are easier to pursue. By these discussions, we hope to convey the idea that ladderpath can provide a different angle when investigating this type of questions such as why life is ordered.

5. Conclusion

In this paper we have argued for the need of a new method for characterizing the complexity of objects that are neither drawn from a statistical distribution nor infinitely large (e.g. infinite strings). We have presented the ladderpath approach which decomposes a single finite object into a partially ordered multiset which describes how the object can be formed by joining or altering existing building blocks. From the minimal ladderpath we can compute two measures of information, namely the ladderpath-index and the order-index, which capture two distinct dimensions of complexity: the difficulty of generating the object and the degree of order. This framework forces a distinction between isolated and united systems, and also puts our understanding of the origin of life and evolution into a clearer view. Here we only scratch the surface of many possible areas of application and it is our hope that this approach will be successfully applied in fields such as linguistics, technological evolution, and exobiology.

Author Contributions: Conceptualization, Y.L.; methodology, Y.L.; software, Y.L.; validation, Y.L. and P.G.; formal analysis, Y.L. and P.G.; investigation, Y.L. and P.G.; resources, Y.L., Z.D. and P.G.; data curation, Y.L.; writing—original draft preparation, Y.L., Z.D. and P.G; writing—review and editing, Y.L., Z.D. and P.G.; visualization, Y.L.; supervision, Y.L. and P.G.; project administration, Y.L.; funding acquisition, Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: The author Y.L. thanks the start-up funding for research (no. 28705-310432106) provided by Beijing Normal University.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: There is no data for this paper. The code is available at https://github.com/yuernestliu/ladderpath

Acknowledgments: We thank Prof. Da Zhou and Prof. Xiaofeng Cai for fruitful discussions, and the five reviewers for their insightful comments and suggestions.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A: Glossary

	Name:	Refer to:	Description or definition:
	Basic set	Section 2.2	The *partially ordered multiset* that contains all the basic elements that constitute an object, denoted as $\mathbf{S}_0$.
	Level	Section 2.2	It refers to the partial order in the *partially ordered multiset*. Each level is separated by double slash "//".
⋆	(Building) block	Section 2.2	Any element in the *partially ordered multiset*.
	Basic (building) block	Section 2.2	Any element in the basic set $\mathbf{S}_0$.
	Target (building) block	Section 2.2	The object to generate in the end, denoted as $\mathcal{X}$.
⋆	Generation-operation	Section 2.2	A specific operation applied on the *partially ordered multiset*. Details in the main text.
⋆	Ladderpath	Section 2.3	A sequence of generation-operations that generate the target block $\mathcal{X}$ in the end, which can be represented by a *partially ordered multiset*, denoted as $J_{\mathcal{X}}$. Details in the main text.
⋆	Ladderon	Section 2.3	Any block in the ladderpath is called a ladderon.
	Trivial ladderpath	Section 2.3	The particular ladderpath(s) in which all of the ladderons are the basic blocks.
⋆	Laddergraph	Section 2.3	A ladderpath can also be represented by a graph or network (besides a *partially ordered multiset*), called a laddergraph.
⋆	Ladderpath-index (λ)	Section 2.4	The length of the shortest ladderpath(s) of an object.
⋆	Size-index (S)	Section 2.4	The length of the shortest trivial ladderpath(s) of an object.
⋆	Order-index (ω)	Section 2.4	Defined as $(S - \lambda)$, referring to Equation (4).
	Length unit of a ladderpath	Section 2.4	A system-dependent quantity. In the string examples in this paper, it is defined as carrying out the action that concatenates any two strings, once.
	Length of a ladderpath	Section 2.4	The sum of the lengths of all generation-operations along the ladderpath.
⋆	*lift*	Section 2.4	The name of the length unit of the ladderpath.
	Shortest ladderpath	Section 2.4	The ladderpath(s) that has the minimum length.
⋆	Target system	Section 2.6	A group of target blocks that need to be generated altogether in the end.
⋆	Ladderpath-system	Section 3	A system that satisfies the two conditions: (i) the ability to generate new blocks, and (ii) some or all blocks can replicate.
	Isolated system	Section 4.1	We call a system an isolated system if we force it to have nothing to do with any other systems.
	Non-isolated (/united) system	Section 4.2	It may not be possible to isolate a system. For example, when considering one single sentence, we should consider it in the context of the language it belongs to, so, this sentence and the language altogether form a united system.

Note that ⋆ represents important concepts.

Appendix B: Examples of the ladderpath of strings

For a better illustration, we give more examples here to show how to construct a ladderpath of a target block, by strictly following the rules elaborated in Section 2.2 and 2.3.

We first show one of the most trivial cases, e.g., string $\mathcal{A}$ = ABCDEF. One ladderpath of $\mathcal{A}$ can be constructed as follows in Ex.B1. In order not to be confused with the set $\mathbf{S}_i$ in the main text, in this appendix we always use $\mathbf{R}_i$. The basic set of $\mathcal{A}$ is $\mathbf{R}_0$ = { A(0), B(0), C(0), D(0), E(0), F(0) }.

Ex.B1. 1st generation-operation, $\mathbf{R}_0$: A+B+C+D+E+F=$\mathcal{A}$ → $\mathbf{R}_1$={ A(-1), B(-1), C(-1), D(-1), E(-1), F(-1) // $\mathcal{A}$(1) }
. The last generation-operation, $\mathbf{R}_1$: $\mathcal{A}$(-1) → $\mathbf{R}_2$={A(-1), B(-1), C(-1), D(-1), E(-1), F(-1) // $\mathcal{A}$(0)}

So, we can see from $\mathbf{R}_2$ that the ladderpath corresponding to Ex.B1 is $J_{\mathcal{A}} = \{\mathrm{A, B, C, D, E, F}\}$. In fact, this is the only ladderpath of $\mathcal{A}$.

The second example we give here is string $\mathcal{B}$ = AAAAAAAA. The basic set of $\mathcal{B}$ is $\mathbf{R}_0$ = { A(0) }. One ladderpath of $\mathcal{B}$ can be constructed as follows:

Ex.B2. 1st generation-operation, $\mathbf{R}_0$: A+A=AA → $\mathbf{R}_1$={ A(-2) // AA(1) }
. 2nd, $\mathbf{R}_1$: AA+AA=AAAA → $\mathbf{R}_2$={ A(-2) // AA(-1) // AAAA(1) }
. 3rd, $\mathbf{R}_2$: AAAA+AAAA=$\mathcal{B}$ → $\mathbf{R}_3$={ A(-2) // AA(-1) // AAAA(-1) // $\mathcal{B}$(1) }
. Lastly, $\mathbf{R}_3$: $\mathcal{B}$(-1) → $\mathbf{R}_4$={ A(-2) // AA(-1) // AAAA(-1) // $\mathcal{B}$(0) }

So, the ladderpath corresponding to Ex.B2 is $J_{\mathcal{B}} = \{\mathrm{A}(2)\ //\ \mathrm{AA}\ //\ \mathrm{AAAA}\}$. Let's construct another ladderpath of $\mathcal{B}$:

Ex.B3. 1st generation-operation, $\mathbf{R}_0$: A+A+A=AAA → $\mathbf{R}_1$={ A(-3) // AAA(1) }
. 2nd, $\mathbf{R}_1$: AAA+AAA=AAAAAA → $\mathbf{R}_2$={ A(-3) // AAA(-1) // AAAAAA(1) }
. 3rd, $\mathbf{R}_2$: A+A+AAAAAA=$\mathcal{B}$ → $\mathbf{R}_3$={ A(-5) // AAA(-1) // AAAAAA(0) // $\mathcal{B}$(1) }
. Lastly, $\mathbf{R}_3$: $\mathcal{B}$(-1) → $\mathbf{R}_4$={ A(-5) // AAA(-1) // AAAAAA(0) // $\mathcal{B}$(0) }

So, the ladderpath corresponding to Ex.B3 is $J'_{\mathcal{B}} = \{\mathrm{A}(5)\ //\ \mathrm{AAA}\}$. After we introduced the definition of the length of a ladderpath in Section 2.4, we can see that the length of $J_{\mathcal{B}}$ is 4 *lifts* while the length of $J'_{\mathcal{B}}$ is 6 *lifts*. In fact, $J_{\mathcal{B}}$ is the shortest ladderpath of $\mathcal{B}$ (which can be confirmed by our algorithm introduced in Section 2.7), and thus the ladderpath-index of $\mathcal{B}$ is $\lambda(\mathcal{B})$= 4 *lifts*.

Another example we give is string $\mathcal{C}$ = ATGTGCATG. Its basic set is $\mathbf{R}_0$ = { A(0), T(0), G(0), C(0) }. One ladderpath of $\mathcal{C}$ can be constructed as follows:

Ex.B4. 1st generation-operation, $\mathbf{R}_0$: T+G=TG → $\mathbf{R}_1$={ A(0), T(-1), G(-1), C(0) // TG(1) }
. 2nd, $\mathbf{R}_1$: A+TG=ATG → $\mathbf{R}_2$={ A(-1), T(-1), G(-1), C(0) // TG(0) // ATG(1) }
. 3rd, $\mathbf{R}_2$: ATG+TG+C+ATG=$\mathcal{B}$ → $\mathbf{R}_3$={ A(-1), T(-1), G(-1), C(-1) // TG(-1) // ATG(-1) // $\mathcal{C}$(1) }
. Lastly, $\mathbf{R}_3$: $\mathcal{C}$(-1) → $\mathbf{R}_4$={ A(-1), T(-1), G(-1), C(-1) // TG(-1) // ATG(-1) // $\mathcal{C}$(0) }

So, the ladderpath corresponding to Ex.B4 is $J_{\mathcal{C}} = \{\mathrm{A, T, G, C}\ //\ \mathrm{TG}\ //\ \mathrm{ATG}\}$. Of course, there are many other ladderpaths for $\mathcal{C}$, but we don't show them here.

Moreover, let's look at string $\mathcal{K}$ = XYXYXYZXYY. Its basic set is $\mathbf{R}_0$ = { X(0), Y(0), Z(0)}. One ladderpath of $\mathcal{K}$ can be constructed as follows:

Ex.B5. 1st generation-operation, $\mathbf{R}_0$: X+Y=XY → $\mathbf{R}_1$={ X(-1), Y(-1), Z(0) // XY(1) }
. 2nd, $\mathbf{R}_1$: XY+XY+XY+Z+XY+Y=$\mathcal{K}$ → $\mathbf{R}_2$={ X(-1), Y(-2), Z(-1) // XY(-3) // $\mathcal{K}$(1) }
. Lastly, $\mathbf{R}_2$: $\mathcal{K}$(-1) → $\mathbf{R}_3$={ X(-1), Y(-2), Z(-1) // XY(-3) // $\mathcal{K}$(0) }

So, the ladderpath corresponding to Ex.B5 is $J_{\mathcal{K}} = \{\mathrm{X, Y}(2), \mathrm{Z}\ //\ \mathrm{XY}(3)\}$.

The last example is string $\mathcal{L}$ = GGGCAUCAUAUAUAUG. Its basic set is $\mathbf{R}_0$ = { A(0), U(0), G(0), C(0) }. One ladderpath of $\mathcal{L}$ can be constructed as follows:

Ex.B6. 1st generation-operation, $\mathbf{R}_0$: A+U=AU → $\mathbf{R}_1$={ A(-1), U(-1), G(0), C(0) // AU(1) }
. 2nd, $\mathbf{R}_1$: C+AU=CAU → $\mathbf{R}_2$={ A(-1), U(-1), G(0), C(-1) // AU(0) // CAU(1) }
. 3rd, $\mathbf{R}_2$: G+G+G+CAU+CAU+AU+AU+AU+G=$\mathcal{L}$ → $\mathbf{R}_3$={ A(-1), U(-1), G(-4), C(-1) // AU(-3) // CAU(-1) // $\mathcal{L}$(1) }
. Lastly, $\mathbf{R}_3$: $\mathcal{L}$(-1) → $\mathbf{R}_4$={ A(-1), U(-1), G(-4), C(-1) // AU(-3) // CAU(-1) // $\mathcal{L}$(0) }

So, the ladderpath corresponding to Ex.B6 is $J_{\mathcal{L}} = \{\mathrm{A, U, G}(4), \mathrm{C}\ //\ \mathrm{AU}(3)\ //\ \mathrm{CAU}\}$.

Appendix C: Compute the ladderpaths of stone patterns

Here we illustrate how to compute the ladderpaths of the stone patterns in Section 2.5. We take the stone pattern [i] and [ii] as two examples, shown in Figure B1a and B1b. The basic set consists of a single stone, and each generation-operation consists of translation, rotation of a block and concatenation with another block. This is illustrated below which strictly follows the definition of ladderpath (see Section 2.3).

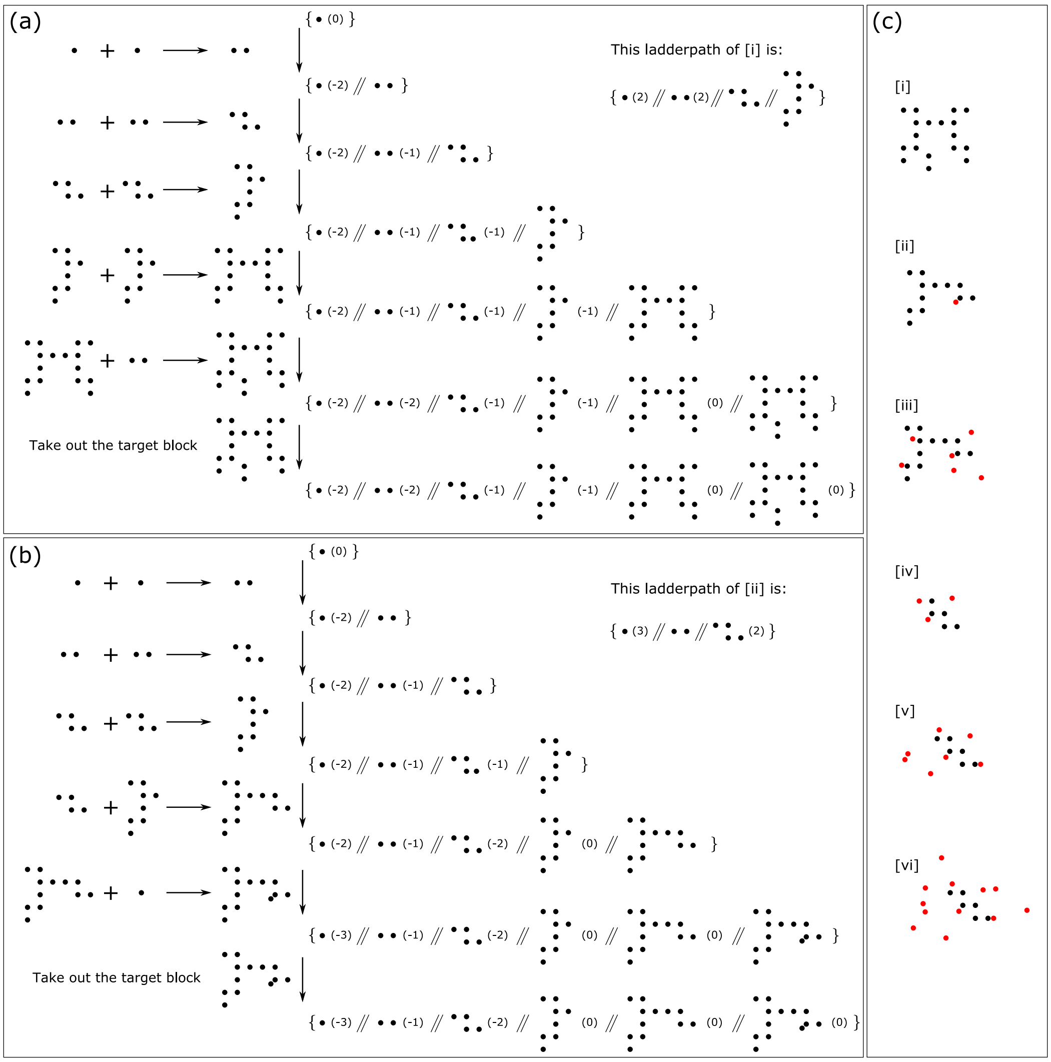

(a)
This ladderpath of [i] is:
Take out the target block
(b)
This ladderpath of [ii] is:
Take out the target block
(c)
[i]
[ii]
[iii]
[iv]
[v]
[vi]

Figure B1. (**a**) The sequence of generation-operations that generate the target stone pattern [i] in the end. Each arrow on the left represents one generation-operation. (**b**) Likewise, the sequence of generation-operations that generate the target stone pattern [ii]. (**c**) It shows the six stone patterns, with the irregular part highlighted. The red color means that these stones are positioned in an irregular manner, and thus, those stones have to be added one by one. For example, for stone pattern [iii], after the black part has been generated, we need six more generation-operations each of which adds one "red" stone.

Appendix D: Example to illustrate the algorithm for the shortest ladderpaths

Here we take the target system $\mathcal{Q}$ = {ABDEDBED(2), ABDED, ABDABD, CAB(2), ED(3)} in Section 2.6 as an example to illustrate the algorithm described in Section 2.7:

i. First create an empty multiset $\mathcal{H}$ to store blocks, and later the ladderpath can be readily computed from $\mathcal{H}$.
ii. Starting from the target system $\mathcal{Q}$, we preserve only one instance of each type of distinct blocks in $\mathcal{Q}$, and put all other repetitions into $\mathcal{H}$. So we have $\mathcal{Q}$ = {ABDEDBED, ABDED, ABDABD, CAB, ED} and $\mathcal{H}$ = {ABDEDBED, CAB, ED(2)} (if there is no bracket behind a block, it means the multiplicity of this block is 1).
iii. And then in the 3rd step, we keep slicing the blocks in $\mathcal{Q}$ in a pre-determined systematic manner, until in $\mathcal{Q}$ there are at least two substrings that are identical. There could be many systematic manners to slice strings, one of which is to treat the positions where slicing could happen as binary numbers and consecutively count. For example, for string *abcd*, there are three positions where slicing could happen, so the consecutive counting of binary numbers is 001, 010, 011, 100, 101, 110, 111, which can be considered as a sequence of an inclusive slicing scheme where 1 denotes for slicing and 0 denotes otherwise. Thus, the sequence of a slicing scheme for *abcd* is as follows: $abc|d$, $ab|cd$, $ab|c|d$, $a|bcd$, $a|bc|d$, $a|b|cd$, $a|b|c|d$.
   In this case, when we count until 0000100, namely slicing ABDEDBED into ABDED and BED, we find that there are two ABDED in $\mathcal{Q}$. So, we put one ABDED into $\mathcal{H}$, and then we have $\mathcal{Q}$ = {BED, ABDED, ABDABD, CAB, ED} while $\mathcal{H}$ = {ABDEDBED, CAB, ED(2), ABDED}.
iv. Repeat step 3. We can slice BED into B and ED, and also ABDED into ABD and ED. Then we find that there are three ED in $\mathcal{Q}$. So, we put two ED into $\mathcal{H}$, and then we have $\mathcal{Q}$ = {B, ABD, ABDABD, CAB, ED} while $\mathcal{H}$ = {ABDEDBED, CAB, ED(4), ABDED}.
v. Repeat step 3. We can slice ABDABD into ABD and ABD. Then we find that there are three ABD in $\mathcal{Q}$. So, we put two ABD into $\mathcal{H}$, and then we have $\mathcal{Q}$ = {B, ABD, CAB, ED} while $\mathcal{H}$ = {ABDEDBED, CAB, ED(4), ABDED, ABD(2)}.
vi. Repeat step 3. We can slice ABD into AB and D, and also CAB into C and AB. Then we find that there are two AB in $\mathcal{Q}$. So, we put one AB into $\mathcal{H}$, and then we have $\mathcal{Q}$ = {B, D, C, AB, ED} while $\mathcal{H}$ = {ABDEDBED, CAB, ED(4), ABDED, ABD(2), AB}.
vii. Repeat step 3. We can slice AB into A and B. Then we find that there are two B in $\mathcal{Q}$. So, we put one B into $\mathcal{H}$, and then we have $\mathcal{Q}$ = {B, D, C, A, ED} while $\mathcal{H}$ = {ABDEDBED, CAB, ED(4), ABDED, ABD(2), AB, B}.
viii. Repeat step 3. We can slice ED into E and D. Then we find that there are two D in $\mathcal{Q}$. So, we put one D into $\mathcal{H}$, and then we have $\mathcal{Q}$ = {B, D, C, A, E} while $\mathcal{H}$ = {ABDEDBED, CAB, ED(4), ABDED, ABD(2), AB, B, D}.
ix. Then we cannot find repetitive letters or strings anymore. Now we put all of the remaining letters into $\mathcal{H}$, and then we have $\mathcal{H}$ = {ABDEDBED, CAB, ED(4), ABDED, ABD(2), AB, B(2), D(2), C, A, E}, while $\mathcal{Q}$ becomes empty.

Now, $\mathcal{H}$ records one ladderpath. Based on how we sliced the strings, we can align the strings in $\mathcal{H}$ in the right hierarchical order, resulting in the partially ordered multiset $\mathcal{H}$ = {A, B(2), C, D(2), E // AB, ED(4) // CAB, ABD(2) // ABDED // ABDEDBED}, namely, a ladderpath (which is actually $J_{\mathcal{Q}}$ in Section 2.6, Equation (7)).

Appendix E: Looking for the evidence of intelligent life

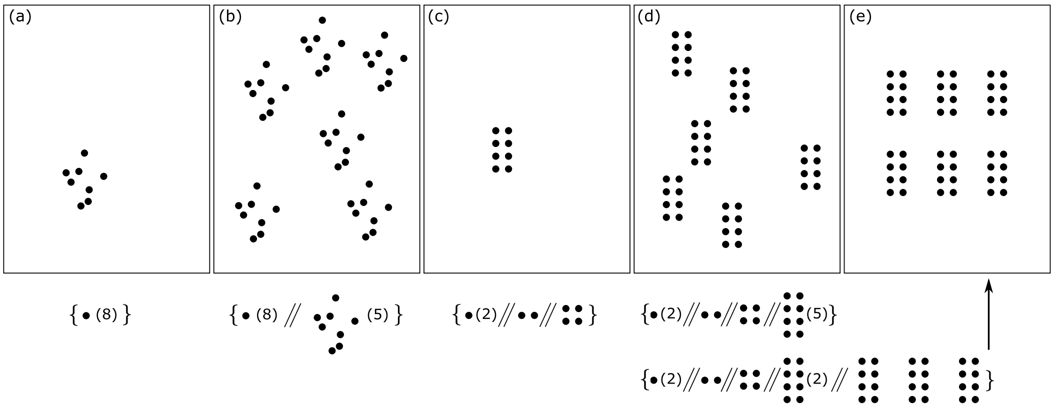
Here we investigate the question of looking for the evidence of intelligent life, which is actually looking at the evolution of ladderpath-systems from the opposite direction, i.e., given the status of a system, we need to tell whether the system reaches this status via the ladderpath mechanism. Here we put forward 5 hypotheses:

1. The order-index ω of life is very high; intelligent life including the things created by intelligent life has an even higher order-index; and the more intelligent, the higher the order-index (although intelligent life can also create things with low order-indices, it

is the upper limit that we are considering here). The reason is that ω describes how different the system is from the systems that are completely generated by random processes, that is, the larger ω is, the more different the system is from random systems; and the involvement of life and intelligent life must deviate the system from randomness.

2. However, the system with a high order-index ω may not always be life, intelligent life, or things created by intelligent life (e.g., crystals produced by simple chemical /physical processes are very much ordered but not life).
3. The ladderpath-index λ (equivalent to the costs or the difficulties to generate /reproduce the system) of life is relatively high; intelligent life including the things created by intelligent life has an even higher ladderpath-index; and the more intelligent, the higher the ladderpath-index (likewise, we only consider the upper limit here). The reason is that, as discussed in the last section, ladderpath-systems automatically evolve more and more "complex" (in both aspects, i.e., increasing order-index and ladderpath-index).
4. However, the system with a high ladderpath-index λ may not always be life, intelligent life, or things created by intelligent life (e.g., the ladderpath-index of a random system is high, but it is not life).
5. The four points above infer that if a system is life, intelligent life, or created by intelligent life, its order-index ω and ladderpath-index λ should be both high, and vice versa (but as for how high can be considered high, that is another question).

We take an example to further explain. As shown in the following figure, imagine that $(a) - (e)$ are five photos of five extraterrestrial planets taken by our spacecraft, where each black dot represents a normal stone. Now we need to tell from these photos whether or not these systems are life, intelligent life, or relics left by intelligent life on the planet. Based on our hypothesis, the most probable one out of the five systems is (b) (as its order-index and ladderpath-index are both very high), and then (d) and (e), but (e) is more likely to be generated by physical processes such as crystallization.



	(a)	(b)	(c)	(d)	(e)
S	8	48	8	48	48
λ	8	13	4	9	7
ω	0	35	4	39	41

where S is the size-index, λ is the ladderpath-index, and ω is the order-index ($:= S - \lambda$). The units of the three quantities are all *lift*.

There are two points to be noticed. (1) It is not certain that (b) must correspond to life or intelligent life, since we do not know how high the values of ω and λ would definitely correspond to life, whereas they can only give us a relative probability. The ladderpath itself cannot tell us the thresholds (if they ever exist) of ω and λ beyond which life or intelligent life definitely exists. Nevertheless, the information of the thresholds might be figured out from other sources, e.g., as Earth is a place with life and intelligent life, we can investigate ω and λ of plant or animal systems, ω and λ of indigenous peoples in isolation, people living in cities, etc., and from those data we may infer the thresholds of ω and λ that life or intelligent life corresponds to (if the thresholds ever exist). (2) When we compare the order-index, we need to set the basic set to be consistent across all systems that we are going to compare, e.g., here we set each basic set to be constituted of the black dots only.

In the example above, we actually considered each photo as an isolated system. Now we consider a different example, namely the following 3 scenarios:

(f) We found an airplane (denoted $\mathcal{F}$) on another planet, which is exactly the same as our airplane on Earth, even the positions of the windows are the same.

(g) We found a UFO-like thing, denoted $\mathcal{G}$, but we have absolutely no idea whether it is an aircraft. It looks like a disc, with a similar size as an airplane, with some patterns on the surface, with a door-like thing but no clue how to get in.

(h) We found two exact $\mathcal{G}$ on another planet.

Now, the question is, which case is most likely to be involved with intelligent life?

In case (f), if $\mathcal{F}$ is considered as an isolated system, its ladderpath can be written as {basic blocks // $\mathcal{F}_{rep}$} (where $\mathcal{F}_{rep}$ is the repetitive part of $\mathcal{F}$), so its order-index is $\omega'_f = S(\mathcal{F}_{rep}) - 1 \doteq S(\mathcal{F}_{rep})$ (according to Equation (6)). However, we should consider $\mathcal{F}$ and airplanes on Earth as a united system since $\mathcal{F}$ is repetitive in our experiences, indeed. So, $\mathcal{F}$ itself is a ladderon and its ladderpath is thus {basic blocks // $\mathcal{F}_{rep}$ // $\mathcal{F}$} (similar to the scenario where we find two identical $\mathcal{F}$), and (f)'s order-index is thus $\omega_f \doteq S(\mathcal{F}_{rep}) + S(\mathcal{F})$. But in case (g), we can only consider $\mathcal{G}$ as an isolated system, since we do not know what it is (thus we cannot consider it and the airplanes on Earth as a united system). So, its ladderpath can only be written as {basic blocks // $\mathcal{G}_{rep}$} (where $\mathcal{G}_{rep}$ is the repetitive parts of $\mathcal{G}$), and its order-index is $\omega_g \doteq S(\mathcal{G}_{rep})$. In case (h), we do not know what $\mathcal{G}$ is neither, but as $\mathcal{G}$ is repetitive in the system itself, its ladderpath is thus {basic blocks // $\mathcal{G}_{rep}$ // $\mathcal{G}$}, and its order-index is $\omega_h \doteq S(\mathcal{G}_{rep}) + S(\mathcal{G})$.

If we further assume that $\mathcal{F}$ and $\mathcal{G}$ have the same size, i.e., $S(\mathcal{F}) = S(\mathcal{G})$, and their repetitive parts also have the same size, i.e., $S(\mathcal{F}_{rep}) = S(\mathcal{G}_{rep})$, then we have $\omega_f = \omega_h > \omega_g$ and $\lambda_f = \lambda_h = \lambda_g$. Then, (f) and (h) are very similar cases, which would be more likely to be caused by intelligent life than (g). Nevertheless, note that if ω_g and λ_g are already very large, that is, the order-index and ladderpath-index of $\mathcal{G}$ are already larger than the "thresholds" that might be caused by intelligent life, inferred from other sources, it is still possible that (g) is also caused by intelligent life.

It is reasonable to hypothesize that not only does the evolution of life follow the mechanism of ladderpath (as explained in Section 3), but also the evolution of civilizations (such as human society and the development of technology). Therefore, when we find or try to find alien civilizations, we might have the following scenarios:

1. On another planet, we find a whole set of processes (or traces of processes) that follow the ladderpath mechanism, while the processes are completely different from what happened on Earth (e.g., silicon-based life, immersed in liquid methane).
2. We try to look for high-λ objects (namely, difficult to make) that look like things in our own world (such as man-made airplanes, architectures with complex shapes but similar to ours, complex metabolic activities but similar to ours, complex machines but similar with ours), which is analogous to case (f). As long as we find a high-λ object similar to the counterpart in our own world (even if we only find one such object), we can treat it as a ladderon (since it is repetitive in this united system). Therefore, both the order-index and the ladderpath-index of this united system are high, which implies that it is the result of a civilization (However, if we and they are independent civilizations, this scenario is very unlikely because the probability that two independent civilizations emerged from literally an infinite number of possible civilizations happen to be identical is practically zero).

3. We find only one single machine (or the relic of one machine), yet it has many repetitive and hierarchical structures inside, resulting in that both its order-index and ladderpath-index are very high. In this case, it is also very likely that this machine was made by an alien civilization.
4. Of course, we cannot exclude the last possibility: we find a large number of identical high-λ objects that have, however, no repetitive structure inside (i.e., each object has a very low order-index). This case is also very likely to be an alien civilization, because on one hand, as the number of objects is very large (the object is thus ladderon), the order-index of the whole system is very large, while on the other hand, as the object has a very high λ, the ladderpath-index λ of the whole system will be also very high. However, the probability that this type of civilization emerges is almost zero (which does not follow the ladderpath mechanism), just as the metaphor of the junkyard tornado, as discussed in Section 4.3.

Appendix F: Connections between the ladderpath and Shannon entropy

First to note is that both the ladderpath and Shannon entropy characterize the syntactic information of an object or system, and have no bearing on the semantic information content. Then, what are the other connections between them? Consider an example, a target system $\mathcal{D}$ which is an enclosed box containing 1 *mol* of butane ($CH_3CH_2CH_2CH_3$) at equilibrium, namely 6.022×10^{23} butane molecules. We know that ladderpath can describe static objects, structures, patterns, etc. So, in this case, we can use the ladderpath to describe a specific point in the state space of these 1 *mol* butane molecules. This state can be considered as a pattern where the gas molecules are all placed randomly /irregularly in space at fixed positions and thus no repetitive pattern arises (referring to Figure 2b, and we could consider those stones as gas molecules). Now, if we define the generation-operation as forming covalent bonds, then the ladderpath of the target system $\mathcal{D}$ is:

$$J_{\mathcal{D}} = \{\mathrm{C}, \mathrm{H}(3) \;//\; \mathrm{CH_2} \;//\; \mathrm{CH_3CH_2} \;//\; \mathrm{CH_3CH_2CH_2CH_3}\ (6.022 \times 10^{23} - 1)\}$$

We can see that $J_{\mathcal{D}}$ mainly describes the molecular structure of butane and the fact that the system $\mathcal{D}$ consists of 1 *mol* butane molecules that are placed randomly.

On the other hand, Shannon entropy of $\mathcal{D}$, reduced to thermodynamic entropy in this case [42], can be calculated from the well-known Sackur-Tetrode equation, which is a function of the molecular internal energy, the number of molecules, and the volume of the container [43]. It can be considered as the description of the logarithm of the number of all possible states constrained by the movements /vibrations /rotations /etc of the molecules. We can see that it is very different from what the ladderpath describes.

Notes

[1] In fact, this string is "*we live in jupiter. we love jupiter*". We just converted a letter into another (including space and period): EMZBRCNKX is "*jupiter.* " (notice period and space), and ACX is "*we.* ", while LPIC is *love*, and LGIC is *live*.

References

1. Smith, J.M. 20th century biology as a science of information (95/102). YouTube Video, 1997. youtube.com/watch?v=78ikxE5-POY.
2. Kolchinsky, A.; Wolpert, D.H. Semantic information, autonomous agency and non-equilibrium statistical physics. *Interface Focus* **2018**, *8*, 20180041. doi:10.1098/rsfs.2018.0041.
3. Crutchfield, J.P.; Young, K. Inferring statistical complexity. *Phys. Rev. Lett.* **1989**, *63*, 105–108. doi:10.1103/PhysRevLett.63.105.
4. Thorén, H.; Gerlee, P. Weak Emergence and Complexity. In Proceedings of the Artificial Life XII Proceedings of the Twelfth International Conference on the Synthesis and Simulation of Living Systems; Fellerman, H.; Dörr, M.; Hanczy, M.; Ladegaard Laursen, L.; Mauer, S.; Merkle, D.; Monnard, P.A.; Støy, K.; Rasmussen, S., Eds.; MIT Press: United States, 2010; pp. 879–886.
5. Kolmogorov, A.N. Three approaches to the quantitative definition of information. *Problemy Peredachi Informatsii [Problems of Information Transmission]* **1965**, *1*, 3–11. doi:10.1080/00207166808803030.
6. Chaitin, G.J. On the length of programs for computing finite binary sequences. *J. ACM* **1966**, *13*, 547–569. doi:10.1145/321356.321363.
7. Chaitin, G.J. On the length of programs for computing finite binary sequences: statistical considerations. *J. ACM* **1969**, *16*, 145–159. doi:10.1145/321495.321506.

8. Chaitin, G.J. *Information-Theoretic Incompleteness*; World Scientific, 1992. doi:10.1142/1861.
9. Zvonkin, A.K.; Levin, L.A. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Mathematical Surveys* **1970**, *25*, 83–124. doi:10.1070/RM1970v025n06ABEH001269.
10. Vitányi, P.M.B. How incomputable is Kolmogorov complexity? *Entropy* **2020**, *22*, 408. doi:10.3390/e22040408.
11. Lempel, A.; Ziv, J. On the Complexity of Finite Sequences. *IEEE Transactions on Information Theory* **1976**, *22*, 75–81. doi:10.1109/TIT.1976.1055501.
12. Ziv, J.; Lempel, A. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory* **1977**, *23*, 337–343. doi:10.1109/TIT.1977.1055714.
13. Ziv, J.; Lempel, A. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory* **1978**, *24*, 530–536. doi:10.1109/TIT.1978.1055934.
14. Adami, C.; Cerf, N.J. Physical complexity of symbolic sequences. *Physica D: Nonlinear Phenomena* **2000**, *137*, 62–69. doi:https://doi.org/10.1016/S0167-2789(99)00179-7.
15. Shannon, C.E. A mathematical theory of communication. *The Bell System Technical Journal* **1948**, *27*, 379–423. doi:10.1002/j.1538-7305.1948.tb01338.x.
16. Grassberger, P. Towards a quantitative theory of self-generated complexity. *International Journal of Theoretical Physics* **1986**, *25*, 907–938. doi:10.1007/BF00668821.
17. Achille, A.; Paolini, G.; Mbeng, G.; Soatto, S. The information complexity of learning tasks, their structure and their distance. *Information and Inference: A Journal of the IMA* **2021**, *10*, 51–72. doi:10.1093/imaiai/iaaa033.
18. Makowski, M.; Piotrowski, E.W.; Frąckiewicz, P.; Szopa, M. Transactional Interpretation for the Principle of Minimum Fisher Information. *Entropy* **2021**, *23*. doi:10.3390/e23111464.
19. Li, M.; Vitanyi, P. *An Introduction to Kolmogorov Complexity and Its Applications*, third ed.; Springer Verlag, 2008.
20. Hansel, G.; Perrin, D.; Simon, I. Compression and entropy. In Proceedings of the STACS 92; Finkel, A.; Jantzen, M., Eds.; Springer Berlin Heidelberg: Berlin, Heidelberg, 1992; pp. 513–528.
21. Ziv, J.; Merhav, N. A measure of relative entropy between individual sequences with application to universal classification. *IEEE Transactions on Information Theory* **1993**, *39*, 1270–1279. doi:10.1109/18.243444.
22. Jacob, F. Evolution and tinkering. *Science* **1977**, *196*, 1161–1166. doi:10.1126/science.860134.
23. Middendorf, M.; Ziv, E.; Wiggins, C.H. Inferring network mechanisms: The Drosophila melanogaster protein interaction network. *Proceedings of the National Academy of Sciences* **2005**, *102*, 3192–3197. doi:10.1073/pnas.0409515102.
24. Wagner, G.P.; Pavlicev, M.; Cheverud, J.M. The road to modularity. *Nature Reviews Genetics* **2007**, *8*, 921–931. doi:10.1038/nrg2267.
25. Valverde, S.; Sole, R. Hierarchical Small Worlds in Software Architecture. In Proceedings of the Software Engineering and Complex Networks, 2007, Vol. 14, *Dynamics of Continuous, Discrete and Impulsive Systems Series B*, pp. 1–11.
26. Valverde, S. Breakdown of Modularity in Complex Networks. *Frontiers in Physiology* **2017**, *8*. doi:10.3389/fphys.2017.00497.
27. Solé, R.; Valverde, S. Evolving complexity: how tinkering shapes cells, software and ecological networks. *Philosophical Transactions of the Royal Society B: Biological Sciences* **2020**, *375*, 20190325. doi:10.1098/rstb.2019.0325.
28. Knuth, D., Evaluation of Powers. In *Art of Computer Programming, Volume 2: Seminumerical Algorithms*, Third ed.; Addison-Wesley Professional, 1997; pp. 461–485.
29. Marshall, S.M.; Murray, A.R.G.; Cronin, L. A probabilistic framework for identifying biosignatures using Pathway Complexity. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* **2017**, *375*, 20160342. doi:10.1098/rsta.2016.0342.
30. Murray, A.; Marshall, S.; Cronin, L. Defining Pathway Assembly and Exploring its Applications. *arXiv* **2018**. doi:10.48550/arXiv.1804.06972.
31. Marshall, S.M.; Moore, D.; Murray, A.R.G.; Walker, S.I.; Cronin, L. Quantifying the pathways to life using assembly spaces. *arXiv* **2019**. doi:10.48550/arXiv.1907.04649.
32. Marshall, S.M.; Mathis, C.; Carrick, E.; Keenan, G.; Cooper, G.J.T.; Graham, H.; Craven, M.; Gromski, P.S.; Moore, D.G.; Walker, S.I.; et al. Identifying molecules as biosignatures with assembly theory and mass spectrometry. *Nature Communications* **2021**, *12*, 3033. doi:10.1038/s41467-021-23258-x.
33. Liu, Y.; Mathis, C.; Bajczyk, M.D.; Marshall, S.M.; Wilbraham, L.; Cronin, L. Exploring and mapping chemical space with molecular assembly trees. *Science Advances* **2021**, *7*, eabj2465. doi:10.1126/sciadv.abj2465.
34. Downey, P.; Leong, B.; Sethi, R. Computing sequences with addition chains. *SIAM Journal on Computing* **1981**, *10*, 638–646. doi:10.1137/0210047.
35. Hordijk, W.; Steel, M. Detecting autocatalytic, self-sustaining sets in chemical reaction systems. *Journal of Theoretical Biology* **2004**, *227*, 451–461. doi:10.1016/j.jtbi.2003.11.020.
36. Liu, Y.; Sumpter, D.J.T. Mathematical modeling reveals spontaneous emergence of self-replication in chemical reaction systems. *Journal of Biological Chemistry* **2018**, *293*, 18854–18863. doi:10.1074/jbc.RA118.003795.
37. Liu, Y. On the definition of a self-sustaining chemical reaction system and its role in heredity. *Biology Direct* **2020**, *15*. doi:10.1186/s13062-020-00269-0.
38. Gatherer, D. Finite universe of discourse: the systems biology of Walter Elsasser (1904-1991). *The Open Biology Journal* **2008**, *1*, 9–20. doi:10.2174/1874196700801010009.
39. Zimmerman, E.; Yonath, A. Biological implications of the ribosome's stunning stereochemistry. *ChemBioChem* **2009**, *10*, 63–72. doi:10.1002/cbic.200800554.

40. Schrödinger, E. *What Is Life? The Physical Aspect of the Living Cell.*; Cambridge University Press, Cambridge., 1944.
41. Wu, K.; Nan, Q.; Wu, T. Philosophical analysis of the meaning and nature of entropy and negative entropy theories. *Complexity* **2020**. doi:10.1155/2020/8769060.
42. Gao, X.; Gallicchio, E.; Roitberg, A.E. The generalized Boltzmann distribution is the only distribution in which the Gibbs-Shannon entropy equals the thermodynamic entropy. *The Journal of Chemical Physics* **2019**, *151*, 034113. doi:10.1063/1.5111333.
43. Schroeder, D.V. *An Introduction to Thermal Physics*; Addison Wesley Longman, 1999.